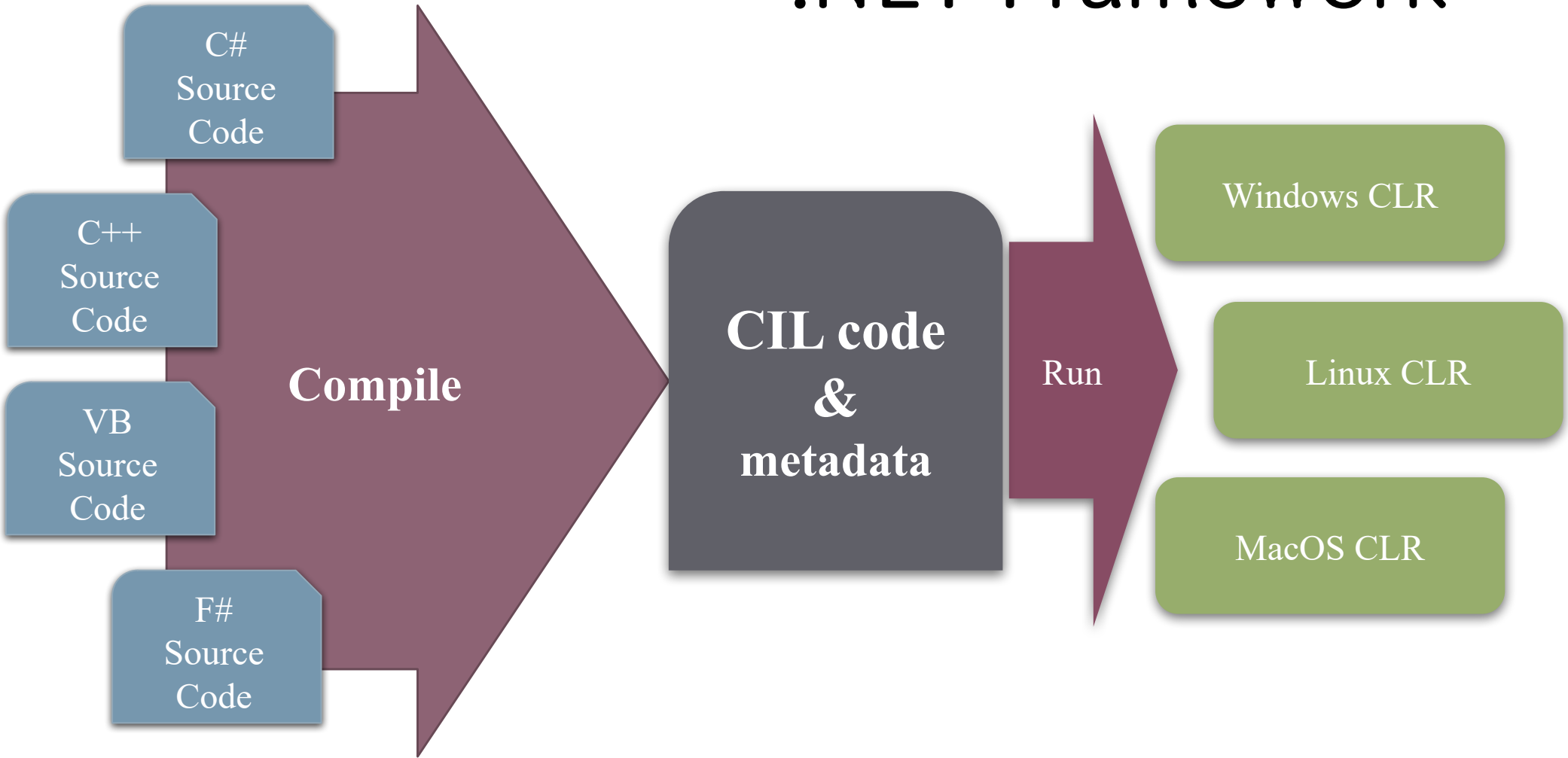
C# 101 Game Developer

Danai Jedsadathitikul

Jarut Busarathid

2025-07-29

.NET Framework



ตัวแปร
และ
ค่าคงที่

การประกาศตัวแปร

Data_type variable_name;

Data_type variable_name = value;

ตัวแปรนั้นเป็นการจองหน่วยความจำของคอมพิวเตอร์เพื่อจัดเก็บตัวเลขลงในหน่วยความจำที่จองไว้ แต่ถ้าหน่วยความจำไม่เพียงพอจะเกิดปัญหา not enough memory ทำให้โปรแกรมไม่สามารถทำงานต่อไปได้

ค่าคงที่

```
const Data_type   constant_name = value;
```

```
const  float  pi = (float)3.1419526;
```

ค่าคงที่เป็นการอ้างอิงชื่อเพื่อใช้แทนค่าตัวเลขทำให้สะดวกในการอ้างอิงหรือเปลี่ยนแปลงค่าแล้วส่งผลให้ทุกชื่อที่อ้างอิงนั้นเปลี่ยนเป็นค่าเดียวกันหมด

Data Types

จำนวนเต็ม → int

ทศนิยม → float หรือ double

bool → true หรือ false

Object → วัตถุต่าง ๆ

DateTime → วันและเวลา

จำนวนที่ใช้กับ C# ประกอบด้วยจำนวนเต็มและทศนิยม โดยจำนวนมีขนาดแตกต่างกันตามตารางที่ 1 เพื่อให้ผู้เขียนโปรแกรมเลือกใช้ตามช่วงค่าที่ต้องการใช้ ทำให้ประหยัดปริมาณหน่วยความจำของคอมพิวเตอร์ หรืออุปกรณ์ที่รันโปรแกรม

ตารางที่ 1-1 ประเภทข้อมูล

จำนวน	ประเภท		ขนาด (ไบต์)	ช่วงค่า	
จำนวนเต็ม	sbyte	System.SByte	1	-128	127
	byte	System.Byte	1	0	255
	bool	System.Boolean	1	False	True
	short	System.Int16	2	-32,768	32,767
	ushort	System.UInt16	2	0	65,535
	int	System.Int32	4	-2,147,483,648	2,147,483,647
	uint	System.UInt32	4	0	4,294,967,295
	long	System.Int64	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
	ulong	System.UInt64	8	0	18,446,744,073,709,551,615
	decimal	System.Decimal	16	-79,228,162,514,264,337,593,543,950,335	79,228,162,514,264,337,593,543,950,335
ทศนิยม	float	System.Single	4	-3.4028235E+38	3.4028235E+38
	double	System.Double	8	-1.7976931348623157E+308	1.7976931348623157E+308

ถ้าเป็นจำนวนเต็มให้ใช้ค่าตัวเลขระบุ เช่น 1, 5, -10, 0 หรือ -200 แต่ถ้าเป็นทศนิยมจะต้องใส่ค่าหลังเลขจำนวนเต็มประกอบ เช่น 1.0, 5.0, -10.0, 0.0 หรือ 200.0 โดยต้องกำหนดอักษร f ต่อท้ายตัวเลขเพื่อระบุประเภทของตัวเลขเป็นทศนิยมแบบ float แต่ถ้าไม่ระบุจำหมายถึง double เป็นต้น

ถ้าเป็นจำนวนเต็มให้ใช้ค่าตัวเลขระบุ เช่น 1, 5, -10, 0 หรือ -200 แต่ถ้าเป็นทศนิยมจะต้องใส่ค่าหลังเลขจำนวนเต็มประกอบ เช่น 1.0, 5.0, -10.0, 0.0 หรือ 200.0 โดยต้องกำหนดอักษร f ต่อท้ายตัวเลขเพื่อระบุประเภทของตัวเลขเป็นทศนิยมแบบ float แต่ถ้าไม่ระบุจำหมายถึง double เป็นต้น

โครงสร้างภาษา

โครงสร้างโปรแกรมพื้นฐานของ C# เป็นดังด้านล่างนี้ โดยจะพบว่า เมื่อใดที่วัตถุถูกเรียกขึ้นใช้งาน และพร้อมทำงานจะมีการเรียกใช้ฟังก์ชัน Main() ให้ทำงานเป็นฟังก์ชันแรกเสมอ และนามสกุลของโค้ดจะเป็น .cs

```
using System;
namespace ชื่อแอปพลิเคชัน
{
        class ชื่อคลาส
        {
                static void Main( string[] args )
                {
                        Console.WriteLine( "Hi, C#!" );
                }
        }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
            Console.ReadLine();
        }
    }
}
```

Console

คลาส Console เป็นคลาสที่อยู่ในเนมสเปซ System มีหน้าที่หลักในการจัดการกับการรับและแสดงผลข้อมูลผ่าน command-line interface (CLI) หรือหน้าต่างคอนโซล เช่น Command Prompt ใน Windows หรือ Terminal ใน Linux/macOS เป็นต้น

คลาส Console มีเมธอดและคุณสมบัติ (properties)

1. **แสดงผลข้อความออกทางหน้าจอ** เป็นการส่งข้อมูลจากโปรแกรมไปแสดงให้ผู้ใช้เห็น
2. **รับข้อมูลจากผู้ใช้** เป็นการอ่านข้อมูลที่ผู้ใช้ป้อนเข้ามาในโปรแกรม
3. **ควบคุมการแสดงผล** เช่น การเปลี่ยนสีตัวอักษร สีพื้นหลัง หรือการเคลียร์หน้าจอ

การแสดงผลข้อความ

Console.WriteLine()

ใช้สำหรับแสดงผลข้อความออกทางหน้าจอ แล้วขึ้นบรรทัดใหม่โดยอัตโนมัติหลังจากแสดงผลเสร็จ

```
// แสดงข้อความ "Hello, World!" และขึ้นบรรทัดใหม่
Console.WriteLine("Hello, World!");
// แสดงตัวเลข 5555 และขึ้นบรรทัดใหม่
Console.WriteLine(5555);
```

```
using System;
namespace Hi
{
    class Program
    {
        static void Main(string[] args)
        {
            var x = 100;
            var y = -50.2;
            const float my_pi = 3.1415f;
            Console.WriteLine("x = " + x);
            Console.WriteLine("y = " + y);
            Console.WriteLine("my_pi = " + my_pi);
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string msg;
            msg = "Hello, World!";
            Console.WriteLine( msg );
            Console.ReadLine();
        }
    }
}
```

```
string msg = "Hello, World!";
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


namespace ConsoleApplication1
{
        class Program
        {
                static void Main(string[] args)
                {
                        string firstName, lastName;
                        Console.WriteLine("Enter your first name : ");
                        firstName = Console.ReadLine();
                        Console.WriteLine("Enter your last name : ");
                        lastName = Console.ReadLine();
                        Console.WriteLine( "Hello, " + firstName + " " + lastName + "!" );
                        Console.ReadLine();
                }
        }
}
```

Console.Write()

ใช้สำหรับแสดงผลข้อความออกทางหน้าจอ แต่จะไม่ขึ้นบรรทัดใหม่หลังจากแสดงผลเสร็จ ทำให้ข้อความถัดไปจะแสดงต่อในบรรทัดเดียวกัน

```
Console.Write("Hello, ");
Console.Write("How are you?");
// ผลลัพธ์: Hello, How are you?
```

Console.WriteAsync() และ Console.WriteLineAsync()

เป็นเวอร์ชัน Asynchronous ของ Write() และ WriteLine() ตามลำดับ เหมาะสำหรับการใช้งานในแอปพลิเคชันที่ต้องการประสิทธิภาพและไม่ต้องการบล็อกเธรดหลักขณะเขียนข้อมูลไปยังคอนโซล

การรับข้อมูลจากผู้ใช้

Console.ReadLine()

ใช้สำหรับอ่านข้อมูลที่ผู้ใช้ป้อนเข้ามาเป็นบรรทัด จนกว่าจะกด Enter และส่งกลับมาเป็นค่า string

```
Console.Write("Name : ");
string name = Console.ReadLine(); // อ่านชื่อที่ผู้ใช้ป้อน
Console.WriteLine("Hello, " + name);
```

Console.Read()

ใช้สำหรับอ่านอักขระตัวแรกที่ผู้ใช้ป้อนเข้ามา และส่งกลับมาเป็นค่า int ซึ่งเป็น ASCII/Unicode value ของอักขระนั้น ๆ ไม่ค่อยได้ใช้บ่อยเท่า ReadLine() เพราะต้องแปลงเป็น char อีกที

```
Console.Write("Press any character : ");
int charCode = Console.Read();
Console.WriteLine("You pressed : " + (char)charCode);
```

Console.ReadKey()

ใช้สำหรับอ่านอักขระตัวแรกที่ผู้ใช้กด โดยไม่ต้องรอการกด Enter และจะส่งคืนค่าเป็นอ็อบเจกต์ ConsoleKeyInfo ซึ่งมีข้อมูลเกี่ยวกับคีย์ที่กด รวมถึง Modifier Keys เช่น Shift, Alt หรือ Ctrl เป็นต้น

Console.ReadKey(true)

อ่านปุ่มที่ผู้ใช้กด ทันที โดยไม่ต้องกด Enter และ true หมายถึงจะไม่แสดงตัวอักษรที่กดบนหน้าจอ

```
Console.Write("Press any key ...");
ConsoleKeyInfo keyInfo = Console.ReadKey();
Console.WriteLine("\nYou pressed : " + keyInfo.Key);
```

ลองแปลง string เป็นตัวเลข

```
string ageString = Console.ReadLine();
int age;
if (int.TryParse(ageString, out age))
{
    //...กรณีที่แปลงสำเร็จ
}
else
{
    // กรณีแปลงผิดพลาด
}
```

การควบคุมการแสดงผล

Console.Clear()

ใช้สำหรับล้างข้อความทั้งหมดที่แสดงอยู่บนหน้าจอคอนโซล

```
Console.WriteLine("Before...");
Console.Clear();
Console.WriteLine("After...");
```

Console.ForegroundColor

คุณสมบัติสำหรับกำหนดสีของตัวอักษรที่แสดงผล โดยค่าสีจะมีค่าเป็น enum ConsoleColor

Console.ResetColor()

คืนค่าสีตัวอักษรและพื้นหลังเป็นค่าเริ่มต้น

```
// เปลี่ยนสีตัวอักษรเป็นสีเขียว
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine("Green text");
// คืนค่าสีตัวอักษรและพื้นหลังเป็นค่าเริ่มต้น
Console.ResetColor();
```

Console.BackgroundColor

คุณสมบัติสำหรับกำหนดสีพื้นหลังของคอนโซล โดยค่าสีจะมีค่าเป็น enum ConsoleColor

```
// เปลี่ยนสีพื้นหลังเป็นสีน้ำเงิน
Console.BackgroundColor = ConsoleColor.Blue;
// เปลี่ยนสีตัวอักษรเป็นสีขาว
Console.ForegroundColor = ConsoleColor.White;
Console.WriteLine("White text on blue background");
Console.ResetColor();
```

Console.Title

คุณสมบัติสำหรับกำหนดชื่อของหน้าต่างคอนโซล

```
Console.Title = "My Game";
```

Console.Beep()

ใช้สำหรับส่งเสียง "Beep" ออกมาทางลำโพงของคอมพิวเตอร์

```
// ส่งเสียง Beep 1 ครั้ง
Console.Beep();
// ส่งเสียง Beep ความถี่ 480 Hz เป็นเวลา 1000 มิลลิวินาที
Console.Beep(480, 1000);
```

Console.CursorVisible = false

ซ่อนเคอร์เซอร์ที่กระพริบ เพื่อให้การแสดงผลดูสะอาดตาขึ้น

Console.SetWindowSize(80, 25)

และ

Console.SetBufferSize(80, 25)

กำหนดขนาดของหน้าต่างและบัฟเฟอร์ของคอนโซล ในที่นี้ตั้งเป็น 80x25 เพื่อให้มีพื้นที่เล่นที่เหมาะสม

Console.SetCursorPosition(x, y)

ย้ายเคอร์เซอร์ไปยังตำแหน่ง (x, y)

Thread.Sleep(x)

ดีเลย์ x มิลลิวินาที เพื่อลดการใช้ CPU

Console.WindowWidth และ Console.WindowHeight

ค่าความกว้างและความสูงของหน้าต่าง

การกำหนดค่า

```
L = R;
```

โดย

L คือ ตัวแปร
R คือ นิพจน์ (expression) เป็นค่าใด ๆ อันอาจจะเป็นตัวเลข ค่าคงที่ ตัวแปร หรือโปรแกรมย่อยที่จะถูกกระทำหรือทำงานจนได้คำตอบเพื่อนำค่านี้นไปเก็บใน L

นิพจน์เป็นการดำเนินการระหว่างตัวดำเนินการตามรูปแบบของเครื่องหมายดำเนินการ อันได้แก่

1. เครื่องหมายดำเนินการ ตัวดำเนินการ
2. ตัวดำเนินการ$_1$ เครื่องหมายดำเนินการ ตัวดำเนินการ$_2$

-a

การนำ -1 **คูณกับ** a

!a

กลับค่าตรรกะของ a **ถ้าเดิม** a **เป็น** true **จะได้ผลลัพธ์เป็น** false **และ ในทางกลับกันเมื่อเดิม** a **เป็น** false **จะได้ผลลัพธ์เป็น** true

not a

กลับค่าตรรกะของ a **ซึ่งให้ผลลัพธ์เหมือนกับการใช้คำสั่ง** !a

เครื่องหมายดำเนินการ

ตารางที่ 1-2 เครื่องหมายดำเนินการทางคณิตศาสตร์

เครื่องหมาย	ความหมาย
*	การคูณ
/	การหาร
%	การหารแบบปัดเศษการหาร
+	การบวก
-	การลบ

true

ตารางที่ 1-3 เครื่องหมายดำเนินการทางตรรกะ

เครื่องหมาย	คำสั่ง	ความหมาย
&&	and	การกระทำแบบ **และ**
||	or	การกระทำแบบ **หรือ**

false

ตารางที่ 1-4 การดำเนินการแบบ **และ**

a	b	a && b
True	True	True
True	False	False
False	True	False
False	False	False

```
using System;
namespace temp_conv
{
    class Program
    {
        public static double C;
        public static double F;
        static void Main(string[] args)
        {
            C = 35.0;
            F = C * 9.0 / 5.0 + 32;
            Console.WriteLine(F);
        }
    }
}
```

$$°F = °C \times (9/5) + 32$$

ตาราง 1-6 เครื่องหมายดำเนินการด้านการเปรียบเทียบ

เครื่องหมาย	ความหมาย
==	ค่าทางซ้ายและขวาเท่ากัน
!=	ค่าทางซ้ายไม่เท่ากับทางขวา
<	ค่าทางซ้ายน้อยกว่าทางขวา
>	ค่าทางซ้ายมากกว่าทางขวา
<=	ค่าทางซ้ายน้อยกว่าหรือเท่ากับทางขวา
>=	ค่าทางซ้ายมากกว่าหรือเท่ากับทางขวา

public

ยินยอมให้เข้าถึงได้จากภายนอกคลาส

static

ให้สร้างตัวแปรนี้ในหน่วยความจำเมื่อมีการประกาศ

public static double C;

ให้ทำการสร้างตัวแปรชื่อ C ให้เป็นตัวแปรประเภททศนิยมที่ใช้พื้นที่ในการจัดเก็บ 8 ไบต์ หลังจากนั้นด้วยคำสั่ง static ทำให้ระบบทำการจองหน่วยความจำตามที่กำหนดเอาไว้ และกำหนดให้สามารถเข้าถึงได้จากนอกคลาส เป็นการสิ้นสุดการสั่งคำสั่ง

Increment/Decrement Operators

++

--

เช่น ++a, a++, --a, a--

++a (1) เพิ่มค่า a
(2) นำ a ไปใช้

a++ (1) นำ a ไปใช้
(2) เพิ่มค่า a

Bitwise Operators

~	complement กลับบิต (1 → 0, 0 → 1)
&	and (1 & 1 → 1, อื่น ๆ → 0)
|	or (0 | 0 → 0, อื่น ๆ → 1)
^	xor (เหมือนกัน → 0, ต่างกัน → 1)

Shift Operators

<<	เลื่อนไปทางซ้าย
>>	เลื่อนไปทางขวา
>>>	???

Assignment

=

+=

-=

*=

/=

%=

<<=

>>=

>>>=

&=

^=

|=

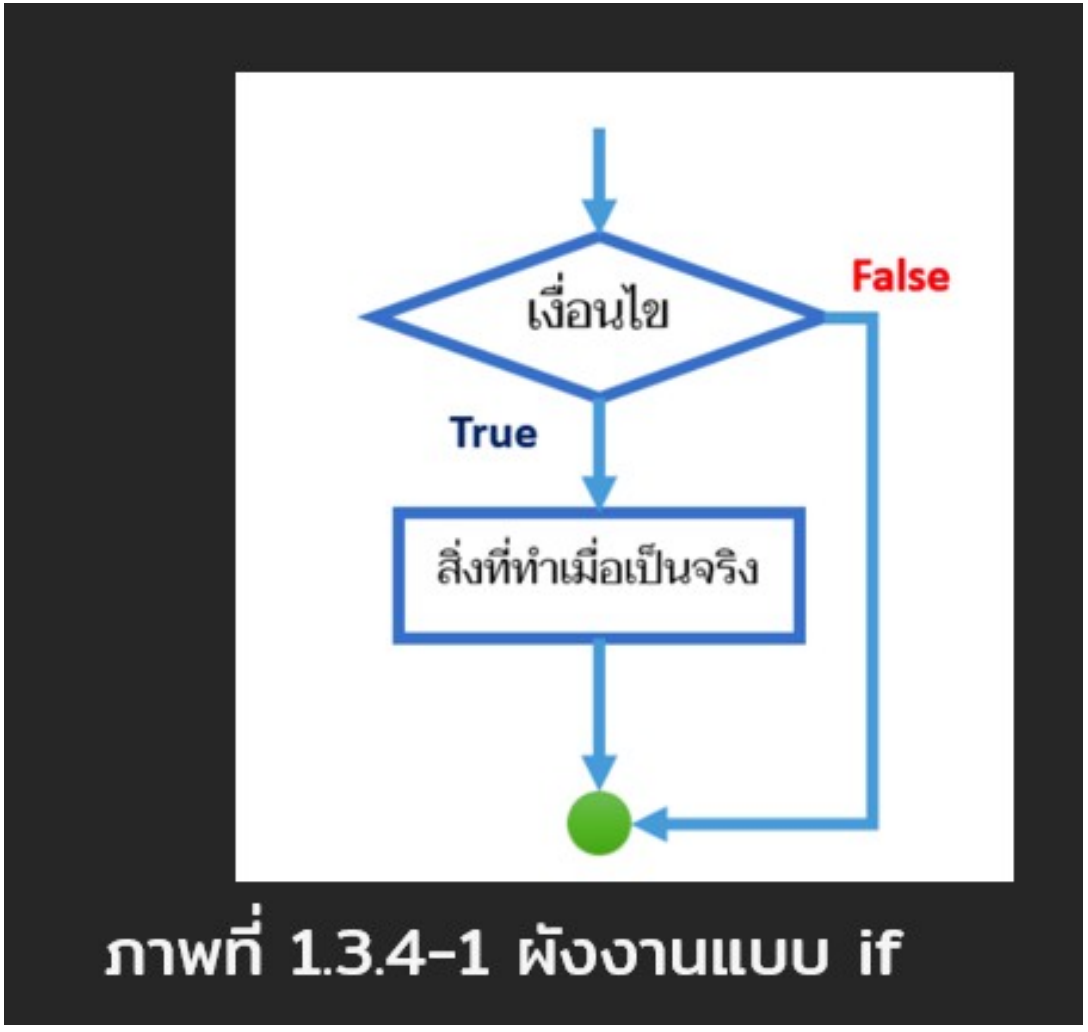
Operator	
Postfix operators	[ ] . (params) expr++ expr--
Unary operators	++expr -expr +expr -expr ~ !
Creation or cast	new (type)expr
multiplication	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >=
Equality	== !=
Bitwise AND	&
Bitwise exclusive OR	^
Bitwise inclusive OR	|
Logical AND	&&
Logical OR	||
conditional	?:
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

เงื่อนไข

การสร้างเงื่อนไขใน C# มีรูปแบบการเขียนด้วยกัน 3 แบบ

ถ้าเงื่อนไขเป็นจริงจะทำตามโค้ดที่เขียน

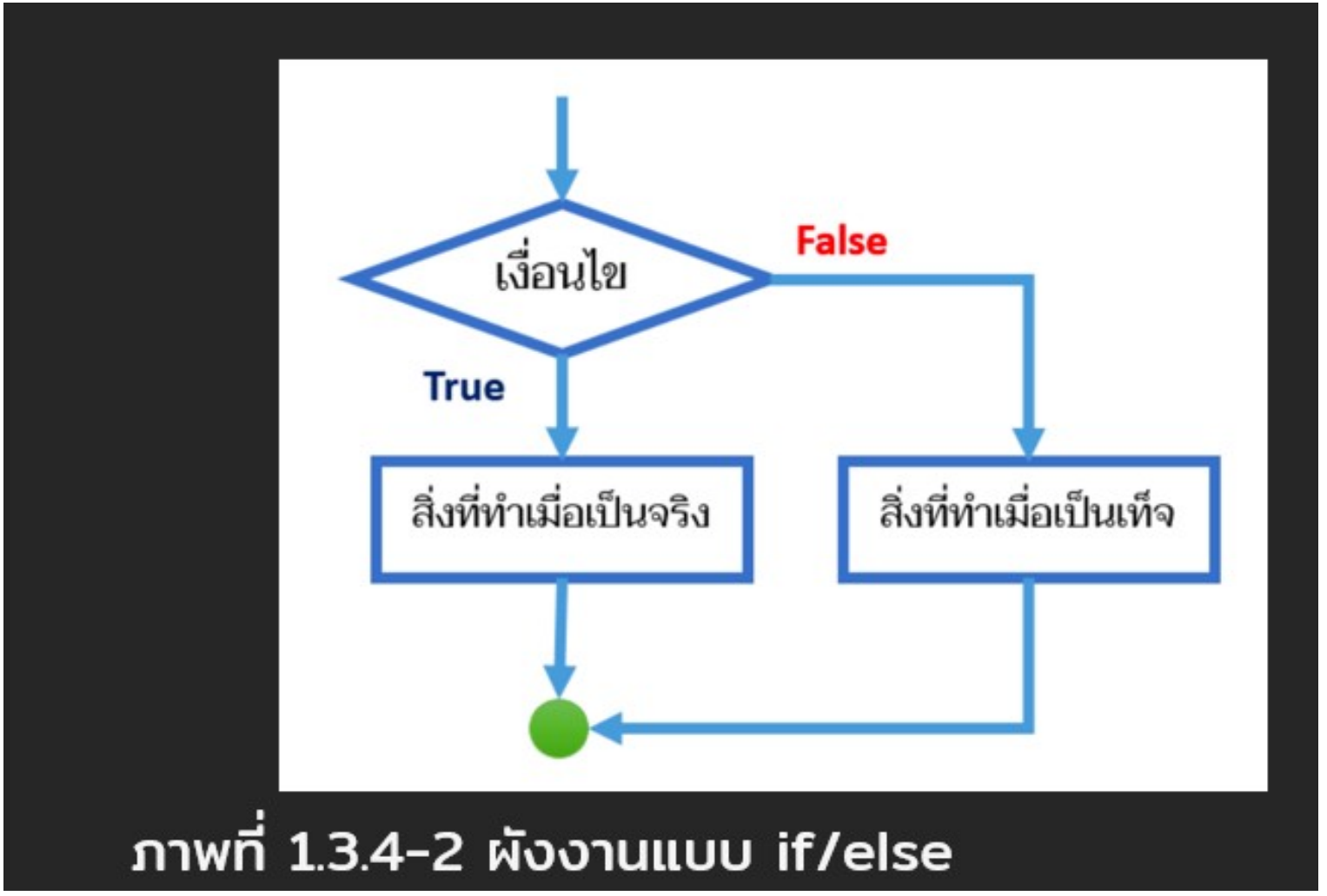
```
if  ( Boolean expression )
{
   // execute this code block if expression is true
}
```



ภาพที่ 1.3.4-1 ผังงานแบบ if

ถ้าเงื่อนไขเป็นจริงจะทำตามโค้ดที่เขียนไว้เพื่อรองรับการทำงานเมื่อไขเป้นจริง แต่ถ้าเงื่อนไขเป็นเท็จจะไปทำงานในส่วนที่อยู่ใน else

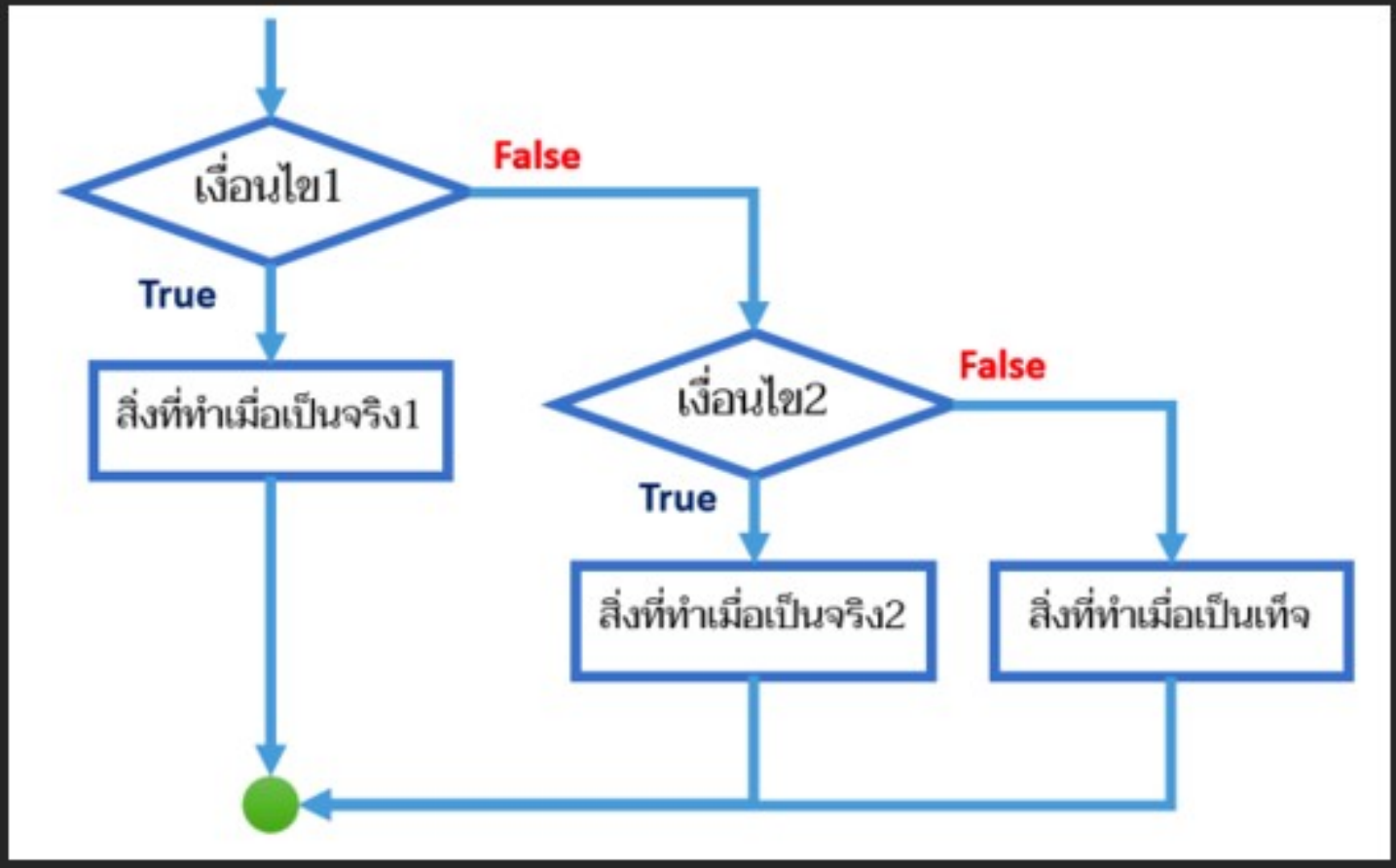
```
if  ( Boolean expression )
{
        // execute this code block if expression is true
}
else
{
        // execute this code block if expression is false
}
```



ภาพที่ 1.3.4-2 ผังงานแบบ if/else

แบบซ้อนเงื่อนไข คือ ทำการตรวจสอบเงื่อนไขภายใต้ if ถ้าเป็นจริงจะทำตามที่เขียนโค้ดเพื่อตอบสนองกรณีที่ if เป็นจริง แต่ถ้าเป็นเท็จจะดำเนินการตรวจสอบเงื่อนไขที่อยู่หลัง else if ถัดไป ถ้าเงื่อนไขเป็นจริงจะทำคำสั่งภายใต้ else if นั้น แต่ถ้าเป็นเท็จจะข้ามไปตรวจสอบ else if ถัดไปเรื่อย ๆ จนไม่พบ else if จะไปทำใน else

```
if ( เงื่อนไข1 )
{
   สิ่งที่ทำเมื่อเงื่อนไข1เป็นจริง
}
else if ( เงื่อนไข2 )
{
   สิ่งที่ทำเมื่อเงื่อนไข2เป็นจริง
}
...
else if ( เงื่อนไขn )
{
   สิ่งที่ทำเมื่อเงื่อนไขnเป็นจริง
}
else
{
   สิ่งที่ทำเมื่อเงื่อนไขเป็นเท็จ
}
```



ภาพที่ 1.3.4-3 ผังงานแบบ if/else ซ้อน

```
using System;
namespace if_else
{
    class Program
    {
        static void Main(string[] args)
        {
            double x = 10.0;
            if (x == 0.0)
            {
                Console.WriteLine(" x == 0 ");
            }
            else
            {
                Console.WriteLine(" x != 0 ");
            }
        }
    }
}
```

```
using System;
namespace level_if
{
    class Program
    {
        public static float score;
        static void Main(string[] args)
        {
            score = 35.0f;
            if (score <10) {
                Console.WriteLine("Level 0");
            } else if (score < 20) {
                Console.WriteLine("Level 1");
            } else if (score < 30) {
                Console.WriteLine("Level 2");
            } else if (score < 40) {
                Console.WriteLine("Level 3");
            } else {
                Console.WriteLine("Level 4");
            }
        }
    }
}
```

```
using System;
namespace err_float2
{
    class Program
    {
        const float epsilon = 0.000001f;
        static void Main(string[] args)
        {
            float a = 0.1f;
            float b = 0.2f;
            float c = a + b;
            Console.WriteLine("a = ", a);
            Console.WriteLine("b = ", b);
            Console.WriteLine("a+b = ", (a + b));
            if ((c - 0.3) < epsilon)
                Console.WriteLine("0.3");

            else
                Console.WriteLine("???");
        }
    }
}
```

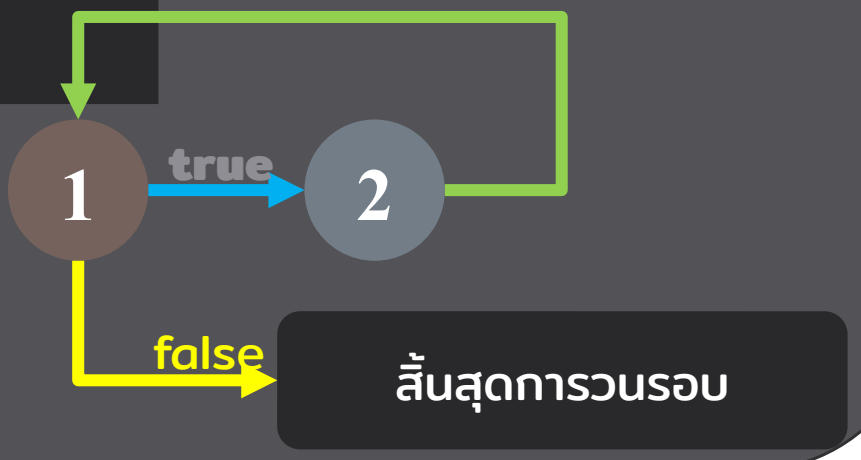
```
switch ( value )
{
        case  value1:
                // execute this code block if value == value1
                break;  // or continue;
        case  value2:
                // execute this code block if value == value2
                break;  // or continue;
        default:
                // execute this code block if nothing true
}
```

การทำซ้ำ

การวนรอบหรือทำซ้ำ (repetition) ใน C# เป็นการใช้ชุดคำสั่งเพื่อให้เกิดการทำซ้ำภายใต้เงื่อนไข เช่น ทำถ้าเป็นจริง และทำเรื่อย ๆ จนกว่าเงื่อนไขที่ตรวจสอบจะเป็นเท็จ หรือ ทำเมื่ออยู่ในช่วงที่กำหนด เป็นต้น

1

```
while   (เงื่อนไข ) {
2     สิ่งที่ทำเมื่อเป็นจริง1
}
```

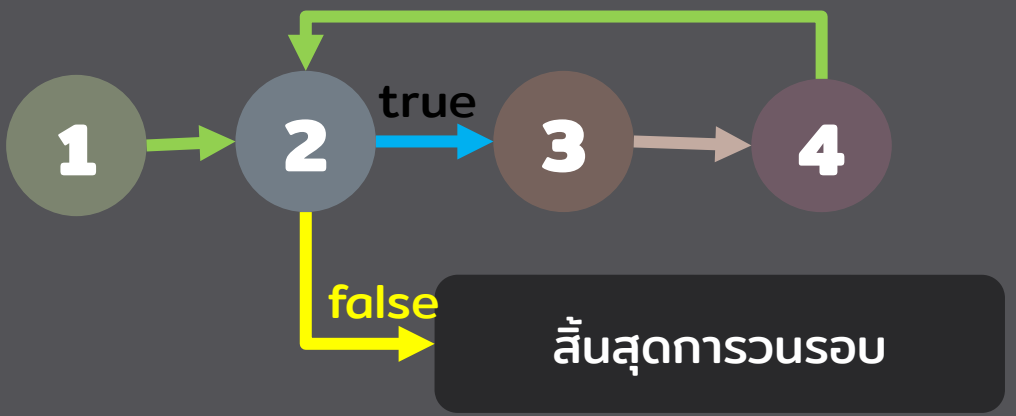


```
using System;
namespace loop_while
{
    class Program
    {
        static Int32 x;
        static void Main(string[] args)
        {
            x = 0;
            while (x <= 12)
            {
                Console.WriteLine(x);
                x = x + 3;
            }
        }
    }
}
```

1 2 4

```
for (ค่าเริ่ม; เงื่อนไข; สิ่งที่ทำก่อนกลับมาตรวจสอบเงื่อนไข) {
    สิ่งที่ทำเมื่อเป็นจริง
}
```

3



```
using System;
namespace loop_for
{
    class Program
    {
        static Int32 x;
        static void Main(string[] args)
        {
            for (x=0; x <= 12; x+=3)
            {
                Console.WriteLine(x);
            }
        }
    }
}
```

```
int j=0;
while (j < 3) {
	System.Console.Beep();
	try
	{
		Thread.currentThread().sleep(500);
	}
	catch (Exception  e)
	{
	}
	j++;
}
```

การวนรอบหรือทำซ้ำโดยทำก่อน 1 รอบ หลังจากนั้นทำการตรวจสอบเงื่อนไข และถ้าเงื่อนไขเป็นจริงจะดำเนินการทำซ้ำอีกครั้งตราบเท่าที่เงื่อนไขนั้นเป้นจริง โดยรูปแบบของการใช้คำสั่งเป็นดังนี้

```
do
{

} while  ( Boolean expression );
```

```
do {
    สิ่งที่ทำเมื่อเป็นจริง
} while (เงื่อนไข );
```

1

2

```
foreach ( data_type var in data)
{

}
```

```
string  []  data = new  string[] {
        “bat”, “rat”, “cat”, “dog”, “pig”
};
foreach ( string animal in data)
{
        Console.WriteLine(“Animal {0}”,animal);
}
```

การสุ่ม

Random เป็นคลาสย่อยของ System

https://learn.microsoft.com/en-us/dotnet/api/system.random?view=net-8.0

```
var rand = new Random();

// Generate and display 5 random byte (integer) values.
var bytes = new byte[5];
rand.NextBytes(bytes);
Console.WriteLine("Five random byte values:");
foreach (byte byteValue in bytes)
{
    Console.Write("{0, 5}", byteValue);
}
Console.WriteLine();
```

```
var rand = new Random();

// Generate and display 5 random integers.
Console.WriteLine("Five random integer values:");
for (int ctr = 0; ctr <= 4; ctr++)
{
    Console.Write("{0,15:N0}", rand.Next());
}
Console.WriteLine();
```

```
var rand = new Random();


// Generate and display 5 random integers between 0 and 100.
Console.WriteLine("Five random integers between 0 and 100:");
for (int ctr = 0; ctr <= 4; ctr++)
{
    Console.Write("{0,8:N0}", rand.Next(101));
}
Console.WriteLine();
```

```
var rand = new Random();
var bytes = new byte[5];

rand.NextBytes(bytes);
Console.WriteLine("Five random byte values:");
foreach (byte byteValue in bytes)
{
    Console.Write("{0,5}", byteValue);
}
Console.WriteLine();
```

```
var rand = new Random();



// Generate and display 5 random floating point values from 0 to 1.
Console.WriteLine("Five Doubles.");
for (int ctr = 0; ctr <= 4; ctr++)
{
    Console.Write("{0,8:N3}", rand.NextDouble());
}
Console.WriteLine();
```

```
var rand = new Random();

// Generate and display 5 random floating point values from 0 to 5.
Console.WriteLine("Five Doubles between 0 and 5.");
for (int ctr = 0; ctr <= 4; ctr++)
{
    Console.Write("{0,8:N3}", rand.NextDouble() * 5);
}
```

```
Random rnd = new Random();
string[] malePetNames = { "Rufus", "Bear", "Dakota", "Fido", "Vanya", "Samuel", "Koani",
"Volodya", "Prince", "Yiska" };
string[] femalePetNames = { "Maggie", "Penny", "Saya", "Princess","Abby", "Laila", "Sadie",
"Olivia", "Starlight", "Talla" };
// Generate random indexes for pet names.
int mIndex = rnd.Next(malePetNames.Length);
int fIndex = rnd.Next(femalePetNames.Length);
// Display the result.
Console.WriteLine("Suggested pet name of the day: ");
Console.WriteLine("   For a male:     {0}", malePetNames[mIndex]);
Console.WriteLine("   For a female:   {0}", femalePetNames[fIndex]);
```

Arrays

แถวลำดับ (Array) เป็นโครงสร้างข้อมูลประเภทหนึ่งที่เกิดจากการจองข้อมูลประเภทเดียวกันเป็นจำนวน n ชุด ทำให้ข้อมูลอยู่ต่อเรียงกันไปเป็นลำดับที่ 0 ถึง n-1 อันทำให้สะดวกต่อการบริหารจัดการกับตัวแปร

เช่น ต้องการสร้างตัวแปรเก็บพิกัดค่า x จำนวน 5 ชุด จะต้องสร้างตัวแปร 5 ตัว แต่ถ้าเป็นแถวลำดับจะใช้ตัวแปรตัวเดียว แต่อ้างอิงลำดับของข้อมูลเป็นข้อมูลลำดับที่ 0, 1, 2 ,3 และ 4 เป็นต้น

```
dataType [] arrayName;
```

```
dataType [] arrayName=
    new ชนิดตัวแปร[จำนวนสมาชิก];
```

dataType [จำนวนสมาชิก] arrayName={

ข้อมูล$_0$, ข้อมูล$_1$, ข้อมูล$_2$, ..., ข้อมูล$_{จำนวนสมาชิก-1}$

};

ตัวอย่างการประยุกต์ใช้แถวลำดับเพื่อเก็บค่าสถานะของตัวละคร เช่น ความแข็งแกร่ง (ST: Strength) ความแม่นยำ (AC: Accuracy) ความคล่องแคล่วว่องไว (DX: Dexterity) ความอดทน (RS: Resistance) แต้มชีวิต (HP: Hit Points) แต้มพลังเวทย์มนต์ (MP: Magic Points) และค่าประสบการณ์ (EX: Experience) โดยให้สมาชิกแต่ละตัวเก็บเป็นตัวเลขจำนวนเต็ม และค่าสูงสุดไม่เกิน 4000 พร้อมจัดเก็บแบบเรียงกันไปเป็นลำดับ 0,1,2,3,4 และ 5 สามารถเขียนได้ดังนี้

```
int [7] actor_stat;
```

การเข้าถึงสมาชิกของแถวลำดับจะต้องระบุตำแหน่งลำดับของสมาชิกนั้นดังรูปแบบการใช้งานด้านล่างนี้ โดยสมาชิกตัวแรกเป็นลำดับที่ 0 และลำดับสุดท้ายจะเท่ากับจำนวนสมาชิกลบด้วย 1

```
ชื่อตัวแปร[ ลำดับ ] = ค่า
```

สำหรับการอ้างอิงถึงค่าของข้อมูลในแถวลำดับให้ทำตามรูปแบบต่อไปนี้

ชื่อตัวแปร[ลำดับ]

การตรวจสอบจำนวนสมาชิกภายในแถวลำดับสามารถใช้คำสั่ง Length เพื่ออ่านค่าออกมาตามรูปแบบการใช้งานดังนี้

```
ชื่อตัวแปร.Length
```

การแปลงข้อมูลตัวเลขให้เป็นข้อความมีรูปแบบการใช้งานดังนี้

```
ข้อมูลตัวอักษร = ตัวแปร.ToString()
```

```
using System;
namespace array1
{
    class Program
    {
        static int[] actor_stat;
        static void Main(string[] args)
        {
            actor_stat = new int[7];
            String[] stat_info = {"ST","AC","DX","RS","HP","MP","EX"};
            actor_stat[0] = 5; // ST
            actor_stat[1] = 4; // AC
            actor_stat[2] = 5; // DX
            actor_stat[3] = 5; // RS
            actor_stat[4] = 10; // HP
            actor_stat[5] = 10; // MP
            actor_stat[6] = 0; // EX
            for (int idx=0; idx<actor_stat.Length; idx++)
            {
                Console.WriteLine(stat_info[idx].ToString() +
                                  "="+ actor_stat[idx].ToString());
            }
        }
    }
}
```

สุ่ม 10x12


```
  GNU nano 7.2
using System;
namespace csEx01 {
class MainClass {
  public static double RandD() {
    var rnd = new Random();
    return Convert.ToDouble( rnd.Next(101) )/100.0;
  }
  public static void Main(String [] args) {
    double[,] data=new double[10,12];
    for(int r=0; r<=data.GetUpperBound(0); r++) {
      for (int c=0; c<=data.GetUpperBound(1); c++) {
        data[r,c] = RandD();
        Console.Write("{0:0.0000}", data[r,c]);
      }
      Console.WriteLine();
    }
  }
}
}
```

สุ่ม 10x12


```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class NewBehaviourScript : MonoBehaviour
{
    1 reference
    double RandD()
    {
        var rnd = new System.Random();
        return Convert.ToDouble(rnd.Next(101))/100.0;
    }
    double[,] data = new double[10, 12];
    Unity Message | 0 references
    void Start()
    {
        Debug.ClearDeveloperConsole();
        String msg;
        msg = "";
        for (int r = 0; r <= data.GetUpperBound(0); r++)
        {
            for (int c = 0; c <= data.GetUpperBound(1); c++)
            {
                data[r, c] = RandD();
                msg += $"{data[r, c]:0.00} ";
            }
            msg += "\n";
        }
        Debug.Log(msg);
    }
    Unity Message | 0 references
    void Update()
    {
    }
}
```

```
int [] num = new int[5];
string [,] couples = new  string[3,2];
int [] num2 = new int[5] {1,2,3,4,5};
num[0] = 1;
num[1] = 2;
num[2] = 3;
num[3] = 4;
num[4] = 5;
string [,] dict = new string[3,2]{
    {“bat”,”ค้างคาว”},{“rat”,”หนู”},{“cat”,”แมว”}
}
Console.WriteLine(“{0} = {1}”, dict[0,0], dict[0,1]);
```

จงเขียนโปรแกรมเพื่อสุ่มระเบิดในพื้นที่ขนาด 8x8 โดยให้มีระเบิดจำนวน 10 ลูก พร้อมทั้งคำนวณระบบเดที่อยู่รอบตัวของแต่ละจุด

1	1	1	1	1	1	1	1
1	*	1	1	*	1	1	*
1	1	1	1	2	3	3	2
		1	1	3	*	*	1
1	1	2	*	4	*	4	1
1	*	2	1	3	*	2	
1	1	2	1	2	1	1	
		1	*	1			

*	2						
*	2	1	1	1		1	1
1	2	2	*	1		1	*
	1	*	3	2		1	1
	1	2	*	2	2	2	1
		1	2	4	*	*	1
			1	*	*	3	1
			1	2	2	1	

โปรแกรมย่อย

โปรแกรมย่อย หรือ ฟังก์ชัน (function) หรือ เมธอด (method) เป็นการสร้างกลุ่มของคำสั่งเพื่อนำกลับมาใช้ในภายหลัง อันจะพบว่า ในโค้ดโปรแกรมที่ได้ศึกษามาในตัวอย่างก่อนหน้านี้มีฟังก์ชันหรือโปรแกรมย่อยชื่อ Main() ซึ่งเป็นโปรแกรมย่อยที่ไม่มีการคืนค่ากลับพร้อมทั้งไม่ต้องการพารามิเตอร์ และเป็นฟังก์ชันที่ถูกกำหนดให้เป็นฟังก์ชันแรกที่ถูกเรียกใช้เมื่อโปรแกรมเริ่มทำงาน

ประเภทของโปรแกรมย่อยแบ่งเป็นหลายประเภท ได้แก่

1. โปรแกรมย่อยที่ไม่คืนค่าและไม่ต้องการพารามิเตอร์
2. โปรแกรมย่อยที่ไม่คืนค่าแต่ต้องการพารามิเตอร์
3. โปรแกรมย่อยที่คืนค่าแต่ไม่ต้องการพารามิเตอร์
4. โปรแกรมย่อยที่คืนค่าและต้องการพารามิเตอร์

โปรแกรมย่อยประเภทนี้ไม่ต้องการข้อมูลนำเข้าและไม่คืนค่ากลับ โดยโครงสร้างของโปรแกรมย่อยแบบนี้เป็นดังนี้

```
void ชื่อโปรแกรมย่อย( )
{
        คำสั่ง1;
        คำสั่ง2;
        ...
        คำสั่งn;
}
```

```
void sayHello() {

        Console.WriteLine( “Helo” );

}
```

โปรแกรมย่อยประเภทที่ต้องการข้อมูลนำเข้าแต่ไม่คืนค่ากลับ ซึ่งข้อมูลนำเข้าเหล่านี้ถูกมองเสมือนเป็นตัวแปรภายในของโปรแกรมย่อยทำให้สามารถนำค่ามาใช้หรือเปลี่ยนแปลงค่าได้ตามความต้องการ โดยโครงสร้างของโปรแกรมย่อยแบบนี้เป็นดังนี้

```
void ชื่อโปรแกรมย่อย( ประเภทข้อมูล1 ข้อมูล1,
                       ประเภทข้อมูล2 ข้อมูล2,
                       ...,
                       ประเภทข้อมูลn ข้อมูลn )
{
      คำสั่ง1;
      คำสั่ง2;
      ...
      คำสั่งn;
}
```

```
 void show_add( float x, float y ) {

        Console.WriteLine( x + y );

}
```

โปรแกรมย่อยประเภทไม่ต้องการข้อมูลนำเข้า แต่มีการคืนค่ากลับหลังจากการทำงานเสร็จ เพื่อเปิดโอกาสให้ผู้ใช้งานโปรแกรมย่อยสามารถนำค่าคืนค่ากลับนี้ไปใช้ในการตรวสอบหรือนำไปเก็บในตัวแปรเพื่อนำไปใช้งานต่อไป โดยโครงสร้างของโปรแกรมย่อยแบบนี้เป็นดังนี้

```
ประเภทข้อมูล ชื่อโปรแกรมย่อย( )
    {
        คำสั่ง1;
        คำสั่ง2;
        ...
        คำสั่งn;
      return ค่าคืนกลับ;
    }
```

```
float pi_over_180() {

            return ( 3.1415296f/180.0f );

}
```

โปรแกรมย่อยประเภทต้องการข้อมูลนำเข้า และคืนค่ากลับ ซึ่งข้อมูลนำเข้าเหล่านี้ถูกมองเสมือนเป็นตัวแปรภายในของโปรแกรมย่อยทำให้สามารถนำค่ามาใช้หรือเปลี่ยนแปลงค่าได้ตามความต้องการ พร้อมทั้งเมื่อทำงานเสร็จจะคืนค่ากลับไปให้ระบบ หรือนำไปเก็บในตัวแปรเพื่อเก็บผลลัพธ์ของการทำงานจากโปรแกรมย่อยนี้ต่อไป โดยโครงสร้างของโปรแกรมย่อยแบบนี้เป็นดังนี้

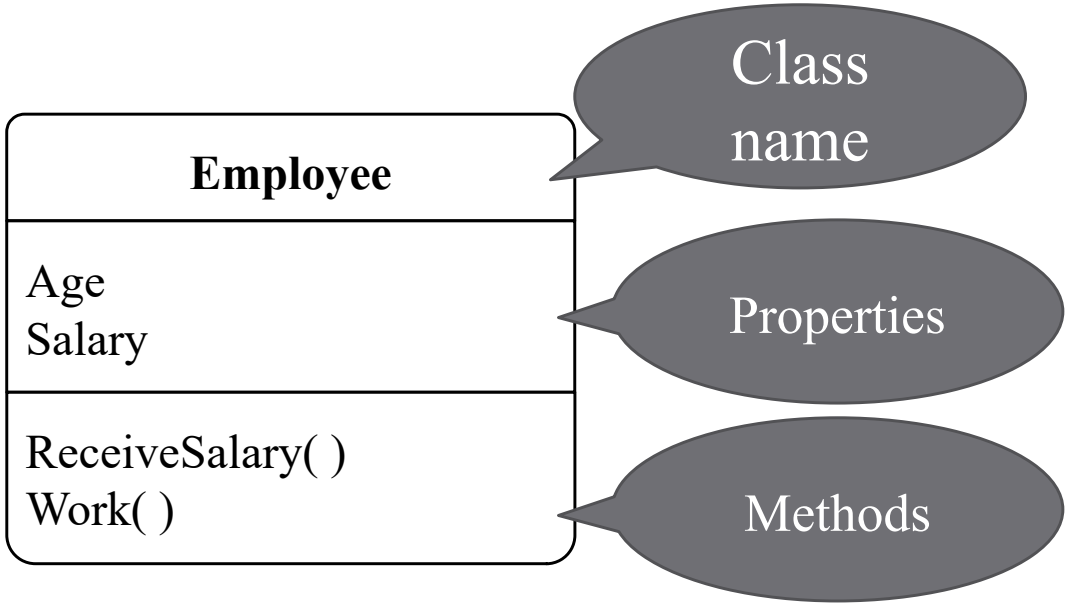
```
ประเภทข้อมูล ชื่อโปรแกรมย่อย(  ประเภทข้อมูล1 ข้อมูล1,
            ประเภทข้อมูล2 ข้อมูล2, ..., ประเภทข้อมูลn ข้อมูลn )
    {
            คำสั่ง1;
            คำสั่ง2;
            ...
            คำสั่งn;
          return ค่าคืนกลับ;
    }
```

```
float degree_to_radian( float degree ) {

            return degree * (180.0f/3.1415926f);

}
```

คลาส




```
class Employee
{
   int  Age;
   double  Salary;
   void  ReceiveSalary()
   {
   }
   Void  Work()
   {
   }
}
```

Constructors

```
public  class Employee
{
    public  int  Age;
    public  double  Salary;
    public  Employee()
    {
    }
    public  Employee(int AgeValue, double SalaryValue)
    {
        Age = AgeValue;
        Salary = SalaryValue;
    }
}
```

Property

```
class Student
{
    private  double  grade;
    public  double  Grade
    {
        get {
            return grade;
        }
        set {
            grade = value
        }
    }
}
```

```
Student  student = new Student();
student.Grade = 3.7;
Console.WriteLine(“Grade: “ + student.Grade);
```

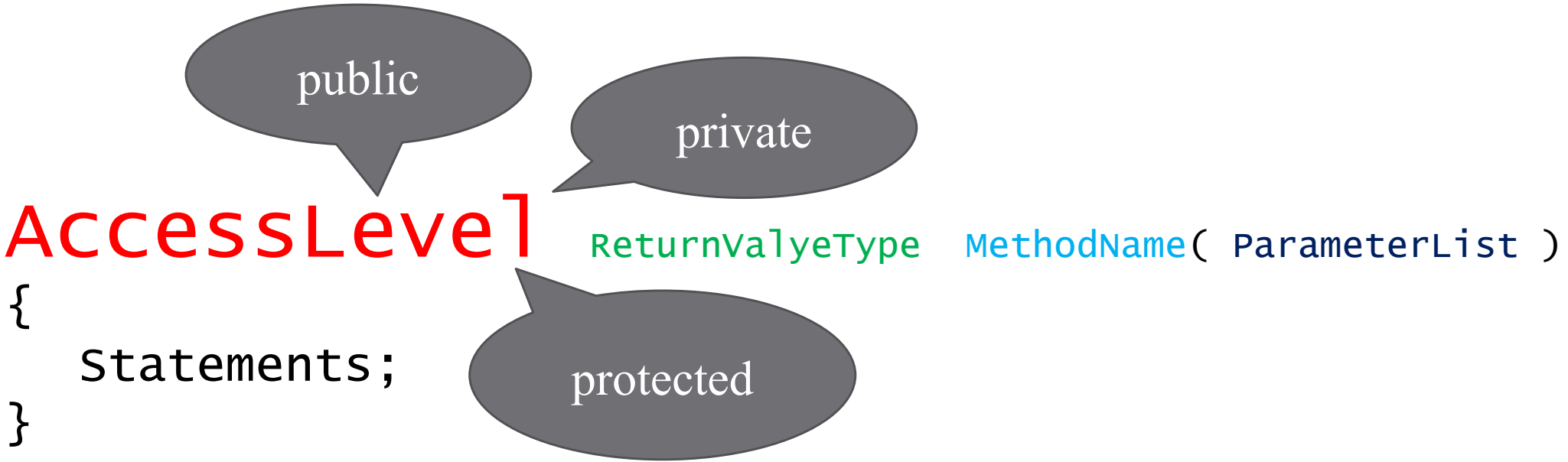
Creating Objects

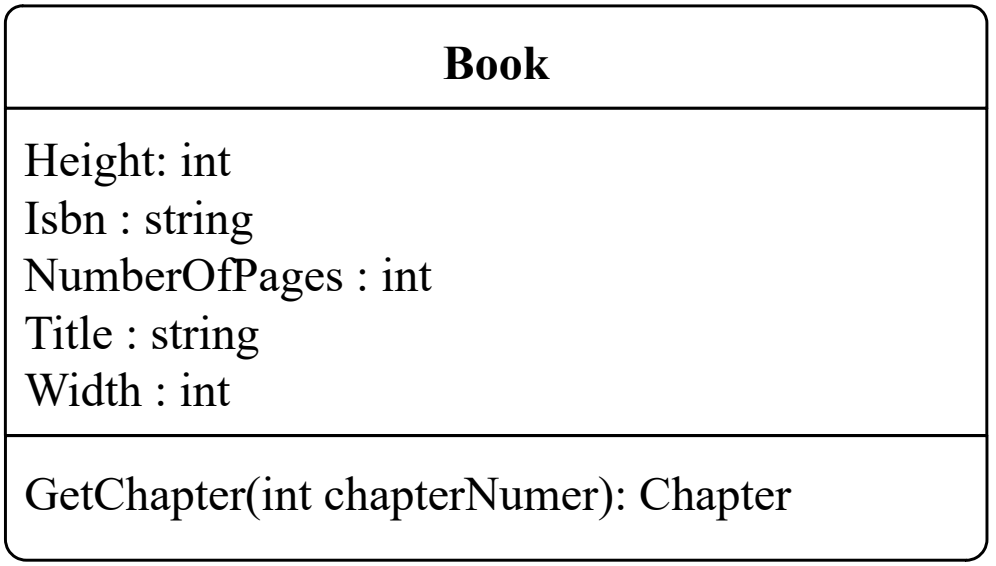
new Object();

```
Employee  employee = new Employee();
```

เมธอด

```
AccessLevel   ReturnValyeType   MethodName( ParameterList )
{
Statements;
}
```





การสืบทอด

```
accessSpecifier  class  baseClass
{
Statements
}

class  derivedClass : baseClass
{
Statements
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Inheritance_Example
{
    class Program
    {
        static void Main(string[] args)
        {
            Eagle b1 = new Eagle();
            b1.EagleInfo();
            Owl b2 = new Owl();
            b2.OwlInfo();
            Chicken b3 = new Chicken();
            b3.ChickenInfo();
            Console.ReadKey();
        }
    }
    class Bird
    {
        protected void BirdInfo()
        {
            Console.WriteLine("It has a pair of wings and a pair of feet.");
            Console.ReadLine();
        }
    }
```

```
    class Eagle:Bird
    {
        public void EagleInfo()
        {
            Console.WriteLine("This is an Eagle, a bird.");
            BirdInfo();
        }
    }
    class Owl:Bird
    {
        public void OwlInfo()
        {
            Console.WriteLine("This is an Owl, a bird.");
            BirdInfo();
        }
    }
    class Chicken:Bird
    {
        public void ChickenInfo()
        {
            Console.WriteLine("This is a chicken, a bird.");
            BirdInfo();
        }
    }
}
```

ตัวอย่าง 1

```
using System;


class Program
{
    static void Main(string[] args)
    {
        // 1. แสดงผลข้อความต้อนรับ
        Console.ForegroundColor = ConsoleColor.Yellow;
        Console.WriteLine("--- Welcone to C# Console ---");
        Console.ResetColor();
        // 2. ขอชื่อจากผู้ใช้
        Console.Write("Name : ");
        string userName = Console.ReadLine();
        // 3. ขออายุจากผู้ใช้
        Console.Write("Age : ");
        string ageString = Console.ReadLine();
        int age;
```

```
// ลองแปลงค่าอายุเป็นตัวเลข
if (int.TryParse(ageString, out age))
{
    Console.WriteLine($"Hello {userName}! You are {age} years old");
    if (age < 18)
    {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("You are a teenager.");
        Console.ResetColor();
    }
    else
    {
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine("You are an adult.");
        Console.ResetColor();
    }
}
```

```
        else
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine("อายุที่คุณป้อนไม่ถูกต้อง.");
            Console.ResetColor();
        }
        // 4. รอให้ผู้ใช้กดปุ่มเพื่อจบโปรแกรม
        Console.WriteLine("\nกดปุ่มใดๆ เพื่อออก...");
        Console.ReadKey();
    }
}
```

ตัวอย่าง 2

ค่าดัชนีมวลกาย =

น้ำหนักตัว [กิโลกรัม] ÷

ส่วนสูง [เมตร] ยกกำลังสอง

น้อยกว่า 18.5 --> Too thin

18.5 - 25.0 --> normal weight

25.0 - 30.0 --> Getting fat

30.0 - 35.0 --> Fat

มากกว่า 35.0 --> Abnormally very fat

```
string firstName, lastName;
long age;
double weight, height, bmi;
Console.Write("Enter your first name : ");
firstName = Console.ReadLine();
Console.Write("Enter your last name : ");
lastName = Console.ReadLine();
Console.WriteLine("Hello, " + firstName + " " + lastName + "!");
Console.Write("Age ? ");
age = Convert.ToInt64(Console.ReadLine());
Console.Write("Weight ? ");
weight = Convert.ToDouble(Console.ReadLine());
Console.Write("Height ? ");
height = Convert.ToDouble(Console.ReadLine())/100.0;
```

```
bmi = weight / (height * height);
if (bmi < 18.5)
{
    Console.WriteLine("too thin");
}
if ((bmi >= 18.5) && (bmi <25.0))
{
    Console.WriteLine("Normal Weight");
}
if ((bmi >= 25.0) && (bmi <30.0))
{
    Console.WriteLine("Getting Fat");
}
if ((bmi >= 30.0) && (bmi <35.0))
{
    Console.WriteLine("Fat");
}
if (bmi >= 35.0)
{
    Console.WriteLine("Abonormally very fat");
}
```

```
bmi = weight / (height * height);
if (bmi < 18.5)
{
    Console.WriteLine("too thin");
}
else if ((bmi >= 18.5) && (bmi <25.0))
{
    Console.WriteLine("Normal Weight");
}
else if ((bmi >= 25.0) && (bmi <30.0))
{
    Console.WriteLine("Getting Fat");
}
else if ((bmi >= 30.0) && (bmi <35.0))
{
    Console.WriteLine("Fat");
}
else if (bmi >= 35.0)
{
    Console.WriteLine("Abonormally very fat");
}
```

ตัวอย่าง 3

จงเขียนโปรแกรมรับข้อมูลส่วนสูงจำนวน 20 ชุด หลังจากนั้นให้รายงาน

-ค่าน้อยสุด

-มากสุด

-ค่าเฉลี่ยของข้อมูล

```
double height, sum = 0.0;
double min = Double.MaxValue;
double max = Double.MinValue;
int loopCounter = 0;
while (loopCounter < 20)
{
    Console.Write("Height ? ");
    height = Convert.ToDouble(Console.ReadLine());
    sum += height;
    if (min > height)
    {
        min = height;
    }
    if (max < height)
    {
        max = height;
    }
    loopCounter++;
}
Console.WriteLine("Mean = {0:0.000}, max={2},min={1}",
    (sum / 20.0), min, max);
```

ตัวอย่าง 4

จงเขียนโปรแกรมรับข้อมูลจำนวนเงิน เพื่อรายงานว่าเงินที่รับเข้ามานั้นแลกเป็นแบงค์/เหรียญใดต่อไปนี้ได้บ้าง

-500

-100

-50

-20

-10

-1

```
double money;
Console.Write("Money ? ");
money = Convert.ToInt64(Console.ReadLine());
Console.WriteLine("B500 = {0}", Convert.ToInt64(money / 500));
money = (money % 500);
Console.WriteLine("B100 = {0}", (money / 100));
money = (money % 100);
Console.WriteLine("B50 = {0}", (money / 50));
money = (money % 50);
Console.WriteLine("B20 = {0}", (money / 20));
money = (money % 20);
Console.WriteLine("C10 = {0}", (money / 10));
money = (money % 10);
Console.WriteLine("C1 = {0}", money);
```

ตัวอย่าง 5

โจทย์ จงเขียนโปรแกรมสุ่มเลขจำนวน 15 ตัว
โดยต้องเป็นเลขในช่วง 1..15 ที่ไม่ซ้ำกัน

```
byte[] bytes1 = new byte[100];
byte[] bytes2 = new byte[100];
Random rnd1 = new Random();
Random rnd2 = new Random();
rnd1.NextBytes(bytes1);
rnd2.NextBytes(bytes2);
Console.WriteLine("First Series:");
for (int ctr = bytes1.GetLowerBound(0);
     ctr <= bytes1.GetUpperBound(0);
     ctr++) {
   Console.Write("{0, 5}", bytes1[ctr]);
   if ((ctr + 1) % 10 == 0) Console.WriteLine();
}
Console.WriteLine();
Console.WriteLine("Second Series:");
for (int ctr = bytes2.GetLowerBound(0);
     ctr <= bytes2.GetUpperBound(0);
     ctr++) {
   Console.Write("{0, 5}", bytes2[ctr]);
   if ((ctr + 1) % 10 == 0) Console.WriteLine();
}
```

```
using System;
using System.Threading;

public class Example
{
   [ThreadStatic] static double previous = 0.0;
   [ThreadStatic] static int perThreadCtr = 0;
   [ThreadStatic] static double perThreadTotal = 0.0;
   static CancellationTokenSource source;
   static CountdownEvent countdown;
   static Object randLock, numericLock;
   static Random rand;
   double totalValue = 0.0;
   int totalCount = 0;

   public Example()
   {
      rand = new Random();
      randLock = new Object();
      numericLock = new Object();
      countdown = new CountdownEvent(1);
      source = new CancellationTokenSource();
   }
```

```
public static void Main()
{
   Example ex = new Example();
   Thread.CurrentThread.Name = "Main";
   ex.Execute();
}
private void Execute()
{
   CancellationToken token = source.Token;
   for (int threads = 1; threads <= 10; threads++)
   {
      Thread newThread = new Thread(this.GetRandomNumbers);
      newThread.Name = threads.ToString();
      newThread.Start(token);
   }
   this.GetRandomNumbers(token);
   countdown.Signal();
   // Make sure all threads have finished.
   countdown.Wait();
   source.Dispose();
   Console.WriteLine("\nTotal random numbers generated: {0:N0}", totalCount);
   Console.WriteLine("Total sum of all random numbers: {0:N2}", totalValue);
   Console.WriteLine("Random number mean: {0:N4}", totalValue/totalCount);
}
```

```
private void GetRandomNumbers(Object o)
{
   CancellationToken token = (CancellationToken) o;
   double result = 0.0;
   countdown.AddCount(1);

   try {
      for (int ctr = 0; ctr < 2000000; ctr++)
      {
         // Make sure there's no corruption of Random.
         token.ThrowIfCancellationRequested();

         lock (randLock) {
            result = rand.NextDouble();
         }
         // Check for corruption of Random instance.
         if ((result == previous) && result == 0) {
            source.Cancel();
         }
         else {
            previous = result;
         }
         perThreadCtr++;
         perThreadTotal += result;
      }
```

```
         Console.WriteLine("Thread {0} finished execution.",
                           Thread.CurrentThread.Name);
         Console.WriteLine("Random numbers generated: {0:N0}", perThreadCtr);
         Console.WriteLine("Sum of random numbers: {0:N2}", perThreadTotal);
         Console.WriteLine("Random number mean: {0:N4}\n", perThreadTotal/perThreadCtr);

         // Update overall totals.
         lock (numericLock) {
            totalCount += perThreadCtr;
            totalValue += perThreadTotal;
         }
      }
      catch (OperationCanceledException e) {
         Console.WriteLine("Corruption in Thread {1}", e.GetType().Name,
Thread.CurrentThread.Name);
      }
      finally {
         countdown.Signal();
      }
   }
}
```

ตัวอย่าง 6

1. ล้างหน้าจอ
2. แสดง o ที่กลางจอภาพ
3. ถ้าผู้เล่นกด a หรือ A ให้ o ขยับไปทางซ้าย แต่ถ้าชนขอบจอให้อยู่ที่เดิม
4. ถ้าผู้เล่นกด d หรือ D ให้ o ขยับไปทางขวา แต่ถ้าชนขอบจอให้อยู่ที่เดิม
5. ถ้าผู้เล่นกด w หรือ W ให้ o ขยับไปด้านบน แต่ถ้าชนขอบจอให้อยู่ที่เดิม
6. ถ้าผู้เล่นกด s หรือ S ให้ o ขยับไปด้านล่าง แต่ถ้าชนขอบจอให้อยู่ที่เดิม
7. ถ้ากด ESC ให้ถามว่าต้องการออกจากโปรแกรมหรือไม่
    7.1 ถ้าไม่ออกให้กลับไปทำ 1
    7.2 ถ้าออกให้จบโปรแกรม
8. กลับไปที่ 1

```
using System;
using System.Threading; // สำหรับ Thread.Sleep()

class Program
{
    static int oX; // ตำแหน่ง X ของตัว 'o'
    static int oY; // ตำแหน่ง Y ของตัว 'o'

    static void Main(string[] args)
    {
        InitializeGame(); // เริ่มต้นเกม: กำหนดตำแหน่งเริ่มต้นของ 'o' และตั้งค่าคอนโซล
        GameLoop();       // วนลูปเกม: รับอินพุตและอัปเดตการแสดงผล
    }
```

```
    static void InitializeGame()
    {
        Console.Title = "ขยับตัว 'o' บนคอนโซล";
        Console.CursorVisible = false; // ซ่อนเคอร์เซอร์กระพริบ
// กำหนดขนาดหน้าต่างคอนโซล (ความกว้าง, ความสูง) ตรวจสอบให้แน่ใจว่าขนาดหน้าต่างที่ตั้งค่าไป
// ไม่เกินขนาดสูงสุดของคอนโซลที่ระบบรองรับถ้าเกิน Console.SetWindowSize จะ throw exception
        int desiredWidth = 80;
        int desiredHeight = 25;
        if (desiredWidth > Console.LargestWindowWidth)
            desiredWidth = Console.LargestWindowWidth;
        if (desiredHeight > Console.LargestWindowHeight)
            desiredHeight = Console.LargestWindowHeight;
// ตั้งค่า buffer size เท่ากับ window size เพื่อป้องกัน scrollbar
        Console.SetWindowSize(desiredWidth, desiredHeight);
        Console.SetBufferSize(desiredWidth, desiredHeight);
// กำหนดตำแหน่งเริ่มต้นของ 'o' ให้อยู่กลางจอ
        oX = Console.WindowWidth / 2;
        oY = Console.WindowHeight / 2;
    }
```

```
static void GameLoop()
{
    bool running = true;
    while (running)
    {
        Console.Clear(); // 1. ล้างหน้าจอ
        DrawO();         // 2. แสดง 'o' ที่กลางจอภาพ (หรือตำแหน่งปัจจุบัน)
        ConsoleKeyInfo key = Console.ReadKey(true); // อ่านปุ่มที่กด (true คือไม่แสดงปุ่มที่กดบนจอ)
        int prevOX = oX; // เก็บตำแหน่งเดิม
        int prevOY = oY;
```

```
            switch (key.Key) // 3-6: จัดการกับการขยับ 'o' ตามปุ่มที่กด
            {
                case ConsoleKey.A: // กด A หรือ a (ซ้าย)
                    oX = Math.Max(0, oX - 1); // ขยับซ้าย แต่ไม่ให้เกินขอบซ้าย (0)
                    break;
                case ConsoleKey.D: // กด D หรือ d (ขวา)
// ขยับขวา แต่ไม่ให้เกินขอบขวา (ความกว้าง-1)
                    oX = Math.Min(Console.WindowWidth - 1, oX + 1);
                    break;
                case ConsoleKey.W: // กด W หรือ w (บน)
                    oY = Math.Max(0, oY - 1); // ขยับขึ้นบน แต่ไม่ให้เกินขอบบน (0)
                    break;
                case ConsoleKey.S: // กด S หรือ s (ล่าง)
// ขยับลงล่าง แต่ไม่ให้เกินขอบล่าง (ความสูง-1)
                    oY = Math.Min(Console.WindowHeight - 1, oY + 1);
                    break;
                case ConsoleKey.Escape: // 7. ถ้ากด ESC
                    running = ConfirmExit(); // ถามว่าต้องการออกจากโปรแกรมหรือไม่
                    break;
            }
```

```
        // ถ้าตำแหน่งเปลี่ยนไป ก็ล้างและวาดใหม่
        if (oX != prevOX || oY != prevOY)
        {
            // ไม่จำเป็นต้อง Console.Clear() ทุกครั้งถ้าเราแค่ลบตัวเก่าและวาดตัวใหม่
            // แต่โจทย์ข้อ 8 บอกให้กลับไปทำ 1 (ล้างหน้าจอ) ทุกรอบ ดังนั้นเราจะใช้ Clear()
            // Console.SetCursorPosition(prevOX, prevOY);
            // Console.Write(" "); // ลบตัว 'o' เก่า
        }

        // สามารถใส่ delay เล็กน้อยเพื่อไม่ให้ CPU ทำงานหนักเกินไป และทำให้การเคลื่อนที่ดูเป็นธรรมชาติขึ้น
        Thread.Sleep(50);
    }
}
```

```
static void DrawO()
{
    // กำหนดตำแหน่งเคอร์เซอร์เพื่อวาด 'o'
    Console.SetCursorPosition(oX, oY);
    Console.ForegroundColor = ConsoleColor.Green; // เปลี่ยนสี 'o' ให้โดดเด่น
    Console.Write("o");
    Console.ResetColor(); // คืนค่าสี
}
```

```
    static bool ConfirmExit()
    {
        Console.Clear(); // ล้างหน้าจอเพื่อแสดงคำถาม
        Console.ForegroundColor = ConsoleColor.Cyan;
        Console.WriteLine("ต้องการออกจากโปรแกรมหรือไม่? (Y/N)");
        Console.ResetColor();
        while (true)
        {
            ConsoleKeyInfo confirmKey = Console.ReadKey(true);
            if (confirmKey.Key == ConsoleKey.Y) // 7.2 ถ้าออกให้จบโปรแกรม
            {
                Console.Clear();
                Console.WriteLine("ออกจากโปรแกรมแล้ว. ขอบคุณที่เล่น!");
                Thread.Sleep(1500); // หน่วงเวลาเล็กน้อยก่อนจบ
                return false; // หยุด GameLoop
            }
            else if (confirmKey.Key == ConsoleKey.N) // 7.1 ถ้าไม่ออกให้กลับไปทำ 1
            {
                return true; // กลับไป GameLoop ต่อ
            }
        }
    }
}
```

Q&A