

1.	ส่วนประกอบของภาษา	3
1.1.	Identifiers	3
1.2.	คำสงวน (Keywords)	3
1.3.	Contextual keywords	4
1.4.	อักขรว่าง Whitespace	4
1.5.	Statements	5
1.6.	Block	5
2.	ตัวแปรและชนิดของข้อมูล Variables and Types	5
2.1.	ข้อมูลชนิดตรรกะ The Boolean Type	6
2.2.	ชนิดข้อมูลตัวเลข (Integer Types)	6
2.3.	ชนิดข้อมูลเลขทศนิยม (Floating Point and Decimal Types)	7
2.4.	ข้อความ (String Type)	7
2.5.	ตัวแปรและค่าคงที่ (Variable and constant)	8
2.6.	Enumeration	8
2.7.	สมการ (Expression)	9
2.8.	คำสั่ง (Statement)	9
3.	การเปลี่ยนทิศทางการทำงานของโปรแกรม	9
3.1.	การเปลี่ยนทิศทางแบบไม่มีเงื่อนไข	9
3.2.	การเปลี่ยนทิศทางแบบมีเงื่อนไข (Conditional Branching)	9
3.2.1.	คำสั่ง if	9
3.2.2.	คำสั่ง if...else	10
3.2.3.	คำสั่ง if ซ้อนกัน (nested if)	11
3.2.4.	คำสั่ง if...else...if	11
3.2.5.	คำสั่ง Switch	12

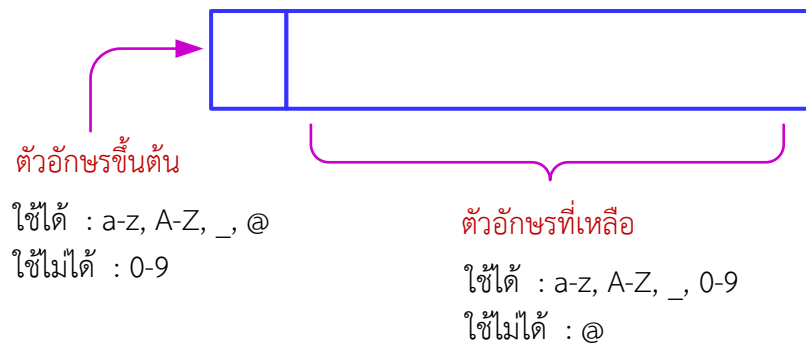
3.3. คำสั่งควบคุมการวนรอบ (Iteration statement)	14
3.3.1. คำสั่ง While	14
3.3.2. คำสั่ง do...while	14
3.3.3. คำสั่ง for	14
3.3.4. คำสั่ง break	15
3.3.5. คำสั่ง continue	16
3.3.6. คำสั่ง foreach...in	16
4. ตัวดำเนินการ (Operators)	17
4.1. ตัวดำเนินการกำหนดค่า (Assignment Operators)	17
4.2. ตัวดำเนินการทางคณิตศาสตร์ (Mathematical Operators)	17
4.3. ตัวดำเนินการเพิ่มค่า/ลดค่า ครั้งละ 1 หน่วย (Incremental/Decremental)	17
4.4. ตัวดำเนินการเพิ่ม/ลดค่า (Calculate & reassign operator)	17
4.5. Prefix/Postfix operator	18
4.6. ตัวดำเนินการเปรียบเทียบ (Relational Operators)	18
4.7. Operator Precedence	19
4.8. The ternary operator	20
5. อื่นๆ / Advanced features	20

1. ส่วนประกอบของภาษา

1.1. Identifiers

Identifiers เป็นอักขระตัวเดียวหรือหลายตัว (strings) ที่ใช้แทนชื่อของตัวแปร เมธอด พารามิเตอร์ หรืออื่นๆ ที่สามารถเก็บค่าและ/หรือเรียกใช้ค่าจากมันในภายหลังได้ โดยทั่วไปในการสร้าง identifier เรามักจะตั้งชื่อให้มีความหมายในตัว (self-documenting identifiers) โดยการนำคำต่างๆ มาสร้างเป็นชื่อของ identifier ที่มีความหมายเป็นเอกลักษณ์ เช่น FirstName, LastName เป็นต้น โดยการตั้งชื่อต้องเป็นไปตามกฎการตั้งชื่อในภาษา C# โดย identifier ประกอบด้วยอักขระต่างๆ ดังนี้

- ตัวอักษรในภาษาอังกฤษ (a ถึง z และ A ถึง Z) รวมทั้งเครื่องหมาย underscore (_) โดยสามารถอยู่ในตำแหน่งใดๆ ก็ได้
- ตัวเลข 0-9 แต่ห้ามใช้เป็นตัวอักษรขึ้นต้น
- เครื่องหมาย @ สามารถใช้เป็นตัวอักษรตัวแรก แต่ห้ามใช้ในตำแหน่งอื่นๆ



รูปที่ 1 การตั้งชื่อ identifier

Identifier จะมีลักษณะที่เรียกว่า case sensitive ดังนั้นต้องระวังในการใช้ตัวอักษรพิมพ์เล็กและพิมพ์ใหญ่ เช่น **myVar** และ **MyVar** จะเป็น identifier คนละตัวกัน

1.2. คำสงวน (Keywords)

คำสงวน ไม่สามารถนำมาใช้เป็นชื่อตัวแปรหรือ identifier ใดๆ ได้ ยกเว้นว่าจะใส่เครื่องหมาย @ ไว้ข้างหน้า โดยคำสงวนทุกคำในภาษา C# จะเป็นตัวพิมพ์เล็กทั้งคำ ยกเว้นชนิดข้อมูลใน .NET จะใช้หลักการตั้งชื่อแบบ Pascal casing ตารางด้านล่างนี้คือคำสงวนในภาษา C#

abstract	const	extern	int	out	short	typeof
as	continue	false	interface	override	sizeof	uint
base	decimal	finally	internal	params	stackalloc	ulong
bool	default	fixed	is	private	static	unchecked
break	delegate	float	lock	protected	string	unsafe
byte	do	for	long	public	struct	ushort
case	double	foreach	namespace	readonly	switch	using
catch	else	goto	new	ref	this	virtual
char	enum	if	null	return	throw	void
checked	event	implicit	object	sbyte	true	volatile
class	explicit	in	operator	sealed	try	while

1.3. Contextual keywords

Contextual keywords จัดเป็น identifiers แต่มีการใช้งานเหมือนคำสงวน เป็นคำที่ใช้ในตำแหน่งเดียวกับคำสงวนของภาษา มันไม่ใช่คำสงวนและไม่สามารถใช้เป็นชื่อหรือ identifier ได้ ดังตารางต่อไปนี้

add	ascending	async	await	by	descending	dynamic
equals	from	get	global	group	in	into
join	let	on	orderby	partial	remove	select
set	value	var	where	yield		

1.4. อักขรว่าง Whitespace

อักขรว่างในโปรแกรมคืออักขรที่มองไม่เห็นเป็นรูปแบบของตัวอักษร (เป็นช่องว่าง) อักขรว่างในโปรแกรมจะถูกมองข้ามโดยคอมไพเลอร์ โดยประกอบด้วยอักขรต่อไปนี้

- Space
- Tab
- New line
- Carriage return

ในภาษา C# นั้น เรามีอิสระที่จะจัดวางอักขรต่างๆ ไว้ที่ตำแหน่งใดก็ได้ โดยคอมไพเลอร์จะพิจารณาการจบบรรทัดด้วยเครื่องหมาย ; และขอบเขตของบล็อกด้วยเครื่องหมายวงเล็บปีกกา { และ }

```
// แบบกระชับ
Main(){Console.WriteLine("Hi, there!");}

// แบบที่อ่านง่ายกว่า
Main()
{
    Console.WriteLine("Hi, there!");
}
```

1.5. Statements

Statements ในภาษา C# จะคล้ายกับในภาษา C และ C++ มีหน้าที่บอกให้โปรแกรมทำงานตามลำดับที่ได้เขียนไว้ตามลำดับ โดย statements ต้องจบด้วยเครื่องหมาย semicolon

```
int var1 = 5;
System.Console.WriteLine("The value of var1 is {0}", var1);
```

1.6. Block

บล็อกอาจจะว่างเปล่าหรือประกอบด้วย statements ตั้งแต่หนึ่ง statement ขึ้นไป โดยขอบเขตของบล็อกเป็นเครื่องหมายวงเล็บปีกกา

```
{
    int var1 = 5;
    System.Console.WriteLine("The value of var1 is {0}", var1);
}
```

2. ตัวแปรและชนิดของข้อมูล Variables and Types

ตัวแปรในภาษาคอมพิวเตอร์ หมายถึงพื้นที่เก็บข้อมูลที่ถูกสร้างขึ้นในหน่วยความจำ เราสามารถใส่ข้อมูลลงไปในตัวแปรและสามารถเรียกใช้ข้อมูลผ่านตัวแปร คุณสมบัติอย่างหนึ่งของตัวแปรคือ ขนาด ซึ่งกำหนดความสามารถในการเก็บข้อมูล โดยทั่วไปตัวแปรจะเก็บข้อมูลที่แตกต่างกันได้เป็นจำนวน 2 ยกกำลังจำนวนบิต ซึ่งเราสามารถสื่อไปยังคอมพิวเตอร์ให้จองพื้นที่ผ่านชนิดข้อมูล (data type) ของตัวแปรนั้นๆ

ภาษา C# เป็นภาษาโปรแกรมแบบ strong type หมายความว่า เราต้องระบุชนิดข้อมูลของตัวแปรก่อนใช้งาน (เช่น integers, floats, strings, windows, buttons, เป็นต้น) โดยชนิดข้อมูลในภาษา C# แบ่งออกได้เป็น 2 กลุ่มคือ ชนิดข้อมูลที่มาพร้อมภาษา (built-in types) เช่น int, bool, long เป็นต้น และชนิดข้อมูลที่เราสร้างเอง (user-defined types) ในภาษา C# ได้แบ่งประเภทการใช้งานตัวแปรออกเป็น 3 รูปแบบได้แก่

1. value type เก็บข้อมูลที่เรียกใช้งานได้ทันที
2. reference type เก็บเฉพาะตำแหน่งที่อยู่ในหน่วยความจำของตัวแปร (หรือวัตถุ)

3. pointer type ซึ่งมีลักษณะเหมือน pointer ในภาษา C หรือ C++ แต่ไม่ค่อยนิยมใช้ เนื่องจากถูกจัดอยู่ในประเภท code ที่ไม่ปลอดภัยสำหรับ .NET (unsafe codes)

ตัวแปรหรือวัตถุในหน่วยความจำ

ตัวแปรหรือวัตถุที่สร้างขึ้นในหน่วยความจำ สามารถเก็บได้ใน 2 แห่งคือ บน stack ของ application หรือบน heap (ซึ่งเป็นหน่วยความจำส่วนรวม) การสร้างตัวแปรบน stack นั้น สามารถสร้างและเรียกใช้งานได้ทันที เนื่องจากระบบได้มอบหน่วยความจำส่วนนั้นให้กับ application ของเราแล้ว เราจะใช้อย่างไรก็ได้ แต่ในส่วน of หน่วยความจำ heap ซึ่งมีการใช้งานร่วมกันทั้งระบบ จะต้องได้รับอนุญาตเสียก่อน จึงจะนำมาใช้ได้ วิธีการใช้งานหน่วยความจำบน heap มีขั้นตอนย่อยๆ อยู่ 4 ขั้นตอนคือ

- 1 สร้างตัวแปร reference ขึ้นบน stack (ของ application)
- 2 จองหน่วยความจำบน heap ด้วยคำสั่ง new แล้วนำตำแหน่งมาเก็บไว้ในตัวแปร reference ในข้อ 1
- 3 ใช้งานหน่วยความจำข้อ 3 (ถ้าจองไม่ได้ ระบบจะให้ค่าตำแหน่ง 0 กลับมา)
- 4 เมื่อใช้งานเสร็จ ให้คืนหน่วยความจำกลับสู่ระบบ

2.1. ข้อมูลชนิดตรรกะ The Boolean Type

ในการสร้างตัวแปร boolean นั้น จะใช้คำว่า bool ซึ่งจะเก็บค่าได้เพียง 2 ค่าเท่านั้นคือ true และ false ในภาษาอื่น เช่น ภาษา C, C++ นั้น สามารถใช้เลข 0 แทน false และเลข 1 แทน true ได้ แต่ในภาษา C# จะไม่สามารถทำได้

ชนิด	ขนาด (บิต)	ขอบเขตการเก็บข้อมูล
bool	8	true และ false

2.2. ชนิดข้อมูลตัวเลข (Integer Types)

ในภาษา C# ชนิดข้อมูลตัวเลข เป็นหมวดของชนิดข้อมูลตัวเลขจำนวนเต็ม ทั้ง signed, unsigned, รวมถึงข้อมูลตัวอักษร (char) ซึ่ง Unicode character

ชนิด	ขนาด (บิต)	ขอบเขตการเก็บข้อมูล
Sbyte	8	-128 ถึง 127
Byte	8	0 ถึง 255
Short	16	-32768 ถึง 32767
ushort	16	0 ถึง 65535

Int	32	-2147483648 ถึง 2147483647
UInt	32	0 ถึง 4294967295
Long	64	-9223372036854775808 ถึง 9223372036854775807
Ulong	64	0 ถึง 18446744073709551615
Char	16	0 ถึง 65535

หมวดของชนิดข้อมูลตัวเลขนี้ สามารถนำมาคำนวณทางคณิตศาสตร์ได้ทุกชนิด (ยกเว้นชนิด char) ในการเลือกใช้งานตัวแปรชนิดต่างๆ ต้องพิจารณาจากขอบเขตของข้อมูลที่เก็บได้เป็นสำคัญ

2.3. ชนิดข้อมูลเลขทศนิยม (Floating Point and Decimal Types)

ชนิดข้อมูลทศนิยมทั้ง float และ double จะถูกใช้แทนจำนวนจริง ซึ่งนิยมใช้ในการคำนวณทางวิทยาศาสตร์หรือในกรณีที่ไม่สามารถเขียนในรูปเศษส่วนที่หารได้ลงตัว ส่วน decimal นิยมใช้ในทางการเงิน (financial). เพราะสามารถหลีกเลี่ยงปัญหาในการปัดเศษ (rounding errors) ได้

ชนิด	ขนาด (บิต)	ความละเอียดทศนิยม (ตำแหน่ง)	ขอบเขต
float	32	7	1.5×10^{-45} ถึง 3.4×10^{38}
double	64	15-16	5.0×10^{-324} ถึง 1.7×10^{308}
decimal	128	28-29	1.0×10^{-28} ถึง 7.9×10^{28}

2.4. ข้อความ (String Type)

ข้อความ (String) เป็นอักขระ (Char) ที่เขียนเรียงต่อกันเป็นข้อความ ในบางกรณีเราต้องการใช้งานตัวอักษรที่ไม่สามารถพิมพ์ออกหน้าจอ (หรือไม่ได้อยู่บนแป้นพิมพ์) ในภาษา C# ก็อนุญาตให้ใช้อักขระเหล่านั้นโดยเขียนนำหน้าด้วยเครื่องหมาย '\' เรียกว่า Escape Sequence

Escape Sequence	ความหมาย
\'	Single Quote
\"	Double Quote
\\	Backslash
\0	Null, คนละอย่างกับ null value ในภาษา C#
\a	Bell
\b	Backspace

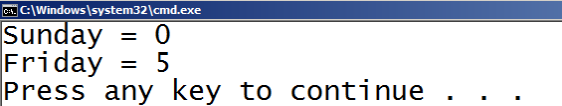
\f	form Feed
\n	Newline
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab

2.5. ตัวแปรและค่าคงที่ (Variable and constant)

ตัวแปรใช้สำหรับเก็บข้อมูลและต้องการที่อยู่ในหน่วยความจำ โดยขนาดหน่วยความจำขึ้นกับชนิดของตัวแปรนั้นๆ ในการใช้งานตัวแปร (Variable) เราสามารถกำหนดค่าได้ทั้งในขณะสร้างและใช้งาน สำหรับค่าคงที่ (constant) เราสามารถกำหนดค่าขณะสร้างแต่ไม่สามารถเปลี่ยนค่าขณะใช้งาน การเลือกใช้งานระหว่างตัวแปรหรือค่าคงที่นั้น ต้องดูว่าต้องการเปลี่ยนค่าในตัวแปรนั้นในขณะใช้งานหรือไม่ เช่น หากต้องการเก็บค่าคงที่ทางวิทยาศาสตร์ ก็ควรเก็บใน constant แทนที่จะเป็นตัวแปร

2.6. Enumeration

Enumerations เป็นวิธีการหนึ่ง ในการใช้ค่าคงที่ได้อย่างมีประสิทธิภาพ โดยการตั้งชื่อเรียกให้กับกลุ่มของค่าคงที่อย่างเป็นระบบ (เรียกว่า enumerator list)

ตัวอย่าง	
<pre>class Program { enum Days { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday }; static void Main(string[] args) { int x = (int)Days.Sunday; int y = (int)Days.Friday; Console.WriteLine("Sunday = {0}", x); Console.WriteLine("Friday = {0}", y); } }</pre>	
Output	
	

2.7. สมการ (Expression)

สมการ หรือ Expression ในภาษา C# คือประโยคที่ใช้ประเมินค่า evaluate ของสมการต่างๆ โดยการกระทำผ่านเครื่องหมาย และจบด้วยเครื่องหมาย semicolon เช่น

```
myVariable = 30;
```

ด้านซ้ายของเครื่องหมาย = เรียกว่า lvalue และด้านขวาของเครื่องหมาย = เรียกว่า rvalue โดย lvalue จะต้องมีสถานะเป็นตัวแปรที่สามารถเก็บข้อมูลได้ ในภาษา C# เราสามารถกำหนดค่าตัวแปรได้เป็นจำนวนมากผ่าน expression เช่น

```
a = b = c = d = e = 30;
```

2.8. คำสั่ง (Statement)

ในภาษา C# เราเรียกคำสั่งที่ทำงานได้โดยสมบูรณ์ว่า statement ซึ่งอาจเป็นประโยคกำหนดค่าหรือประโยคที่เรียกใช้ฟังก์ชันต่างๆ ก็ได้ โดยคำสั่งจะถูกเรียกทำงานตามลำดับจากบนลงล่าง ตามที่เราเขียนลงไป source code

3. การเปลี่ยนทิศทางการทำงานของโปรแกรม

3.1. การเปลี่ยนทิศทางแบบไม่มีเงื่อนไข

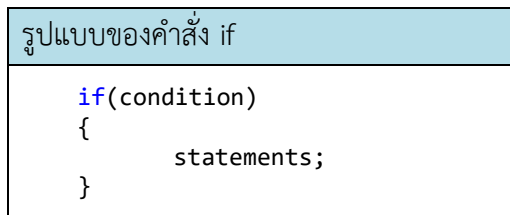
การเปลี่ยนทิศทางการทำงานของโปรแกรมแบบไม่มีเงื่อนไข ทำได้โดยการเรียกใช้เมธอด หรือด้วยการใช้คำสั่ง goto, break, continue, return และ throw

3.2. การเปลี่ยนทิศทางแบบมีเงื่อนไข (Conditional Branching)

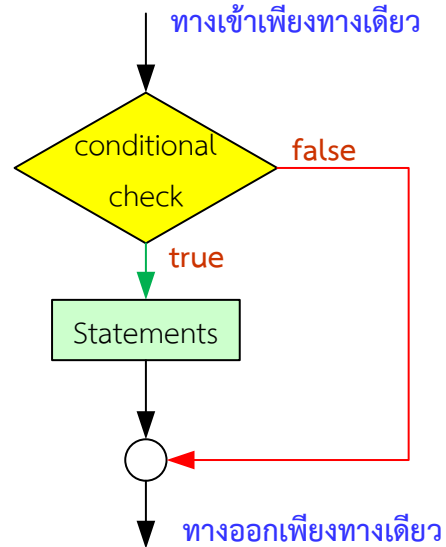
การเปลี่ยนทิศทางแบบมีเงื่อนไขใน C# มีลักษณะคล้ายกับในภาษา C, C++ ประกอบด้วยคำสั่ง if, else และ switch ทั้งนี้ เงื่อนไขที่นำมาตัดสินใจจะต้องมีชนิดเป็น boolean เท่านั้น คำสั่งการเปลี่ยนทิศทางของโปรแกรม จะมีทางเข้าและทางออกเพียงทางเดียว ยกเว้นเสียแต่มีการนำคำสั่งเปลี่ยนทิศทางแบบไม่มีเงื่อนไข เช่น goto และ throw มาไว้ภายใน

3.2.1. คำสั่ง if

คำสั่ง if เป็นคำสั่งที่นำเงื่อนไขมาใช้ในการตัดสินใจเพื่อเปลี่ยนทิศทางการทำงานของโปรแกรม โดยมีรูปแบบดังต่อไปนี้



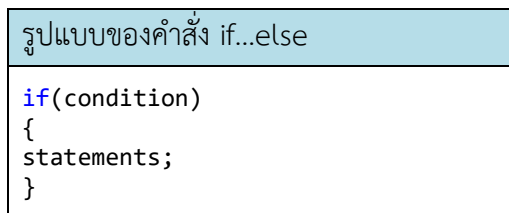
(ก) รูปแบบคำสั่ง



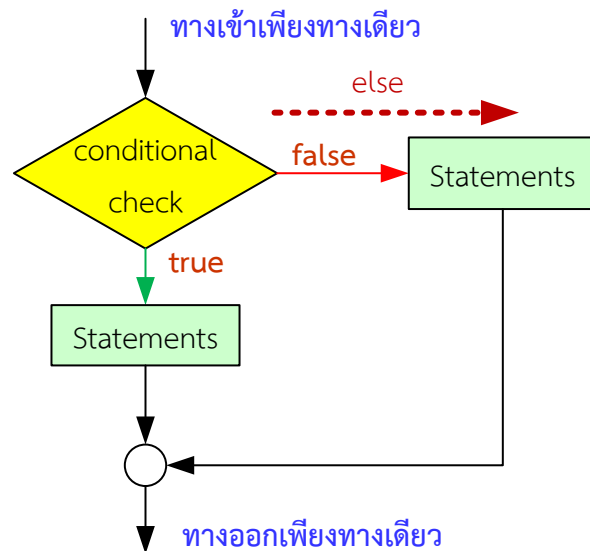
(ข) Flowchart

3.2.2. คำสั่ง if...else

รูปแบบคำสั่งและ Flowchart ของ if...else



(ก) รูปแบบคำสั่ง



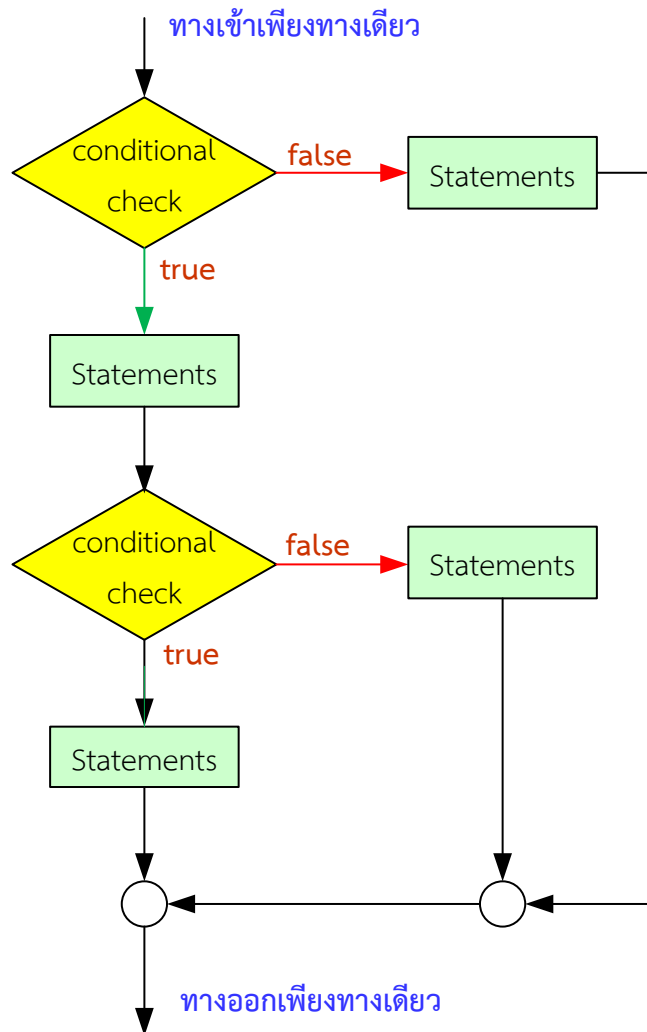
(ข) Flowchart

3.2.3. คำสั่ง if ซ้อนกัน (nested if)

รูปแบบของคำสั่ง nested if

```
if(condition)
{
    statements;
    if(condition)
    {
        statements;
    }
}
```

(ก) รูปแบบคำสั่ง



(ข) Flowchart

3.2.4. คำสั่ง if...else...if

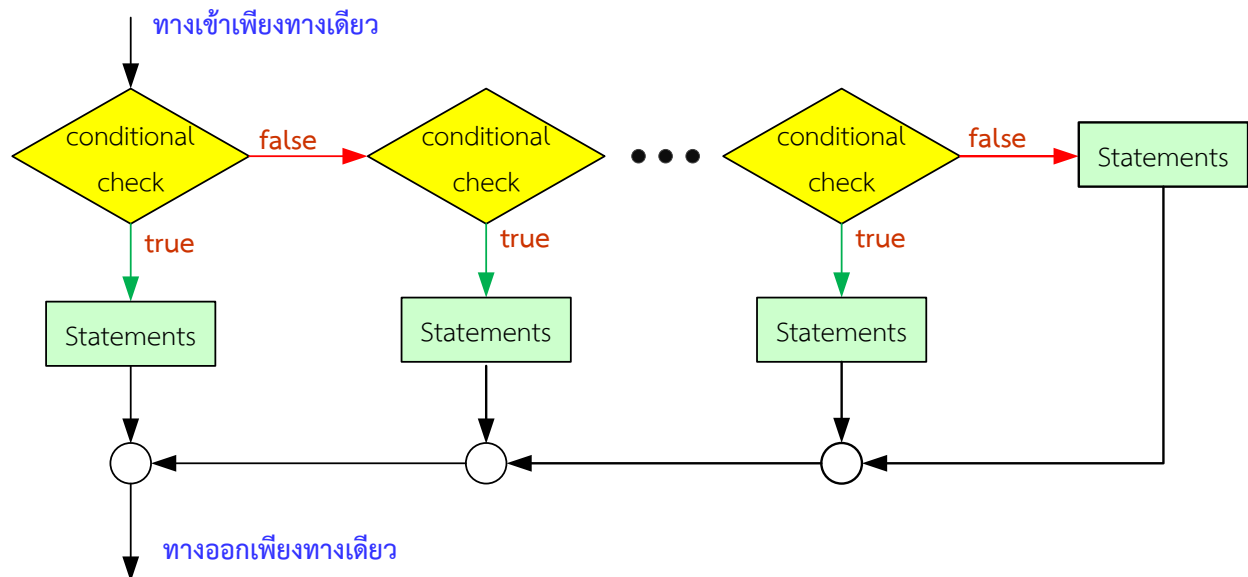
รูปแบบของคำสั่ง if...else...if

```
if (condition)
{
    statements;
}
else if (condition)
{
    statements;
}
...
```

```

else
{
    statements;
}

```



Flowchart ของคำสั่ง if... else...if

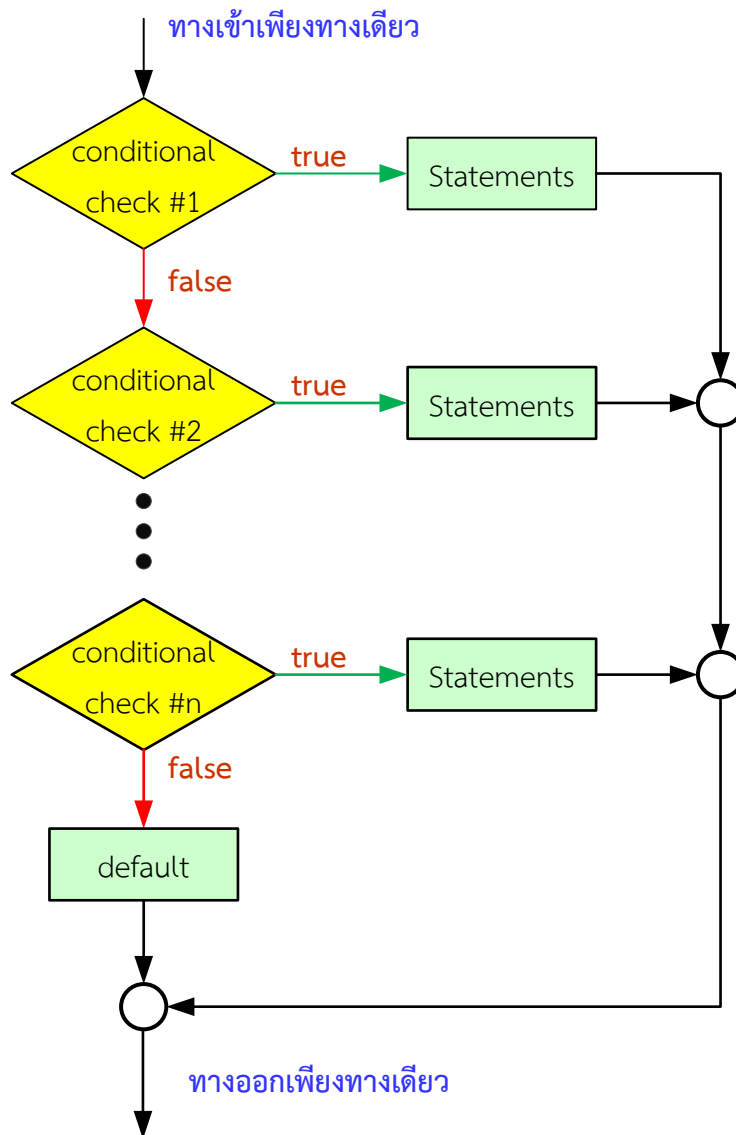
3.2.5. คำสั่ง Switch

รูปแบบของคำสั่ง switch

```

switch (variable)
{
    case constant_1:
        statements;
        break;
    case constant_2:
        statements;
        break;
    ...
    case constant_n:
        break;
    default:
        statements;
        break;
}

```



Flowchart ของคำสั่ง switch

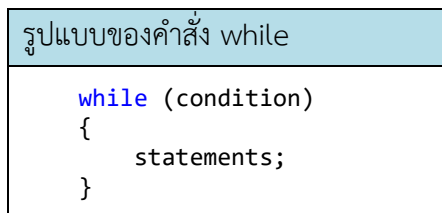
ในภาษา C# เราสามารถใช้ค่าคงที่ที่เป็น string ก็ได้ เช่น

```

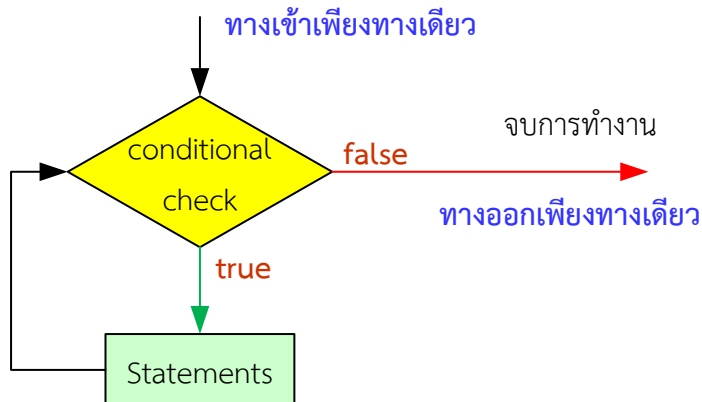
switch (day)
{
    case "Sunday":
        statements;
        break;
    ...
    default:
        statements;
        break;
}
  
```

3.3. คำสั่งควบคุมการวนรอบ (Iteration statement)

3.3.1. คำสั่ง While

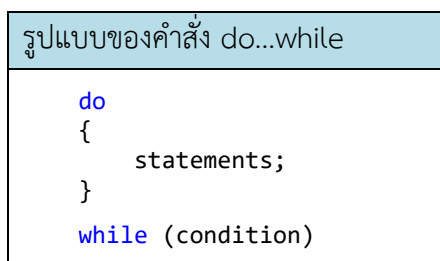


(ก) รูปแบบคำสั่ง

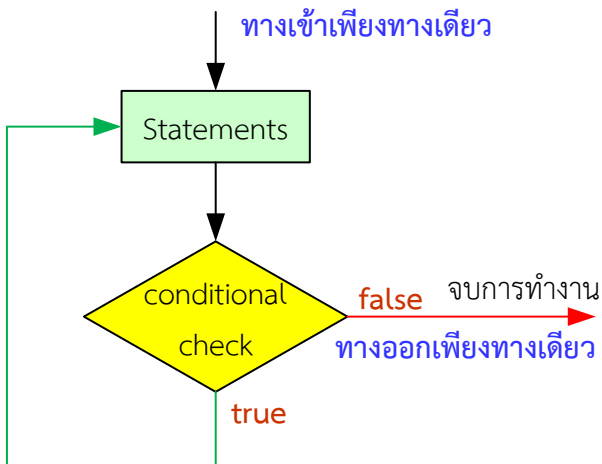


(ข) Flowchart

3.3.2. คำสั่ง do...while

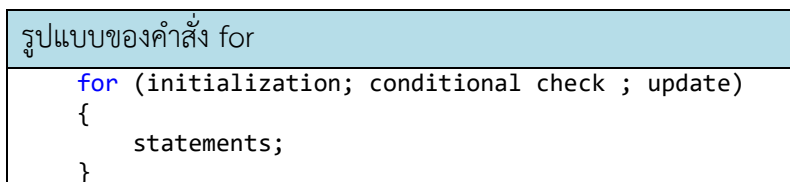


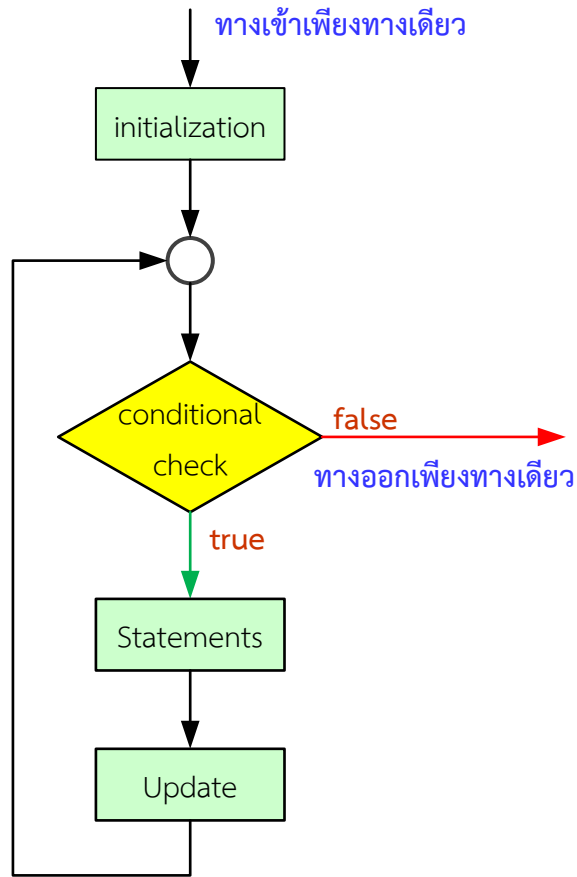
(ก) รูปแบบคำสั่ง



(ข) Flowchart

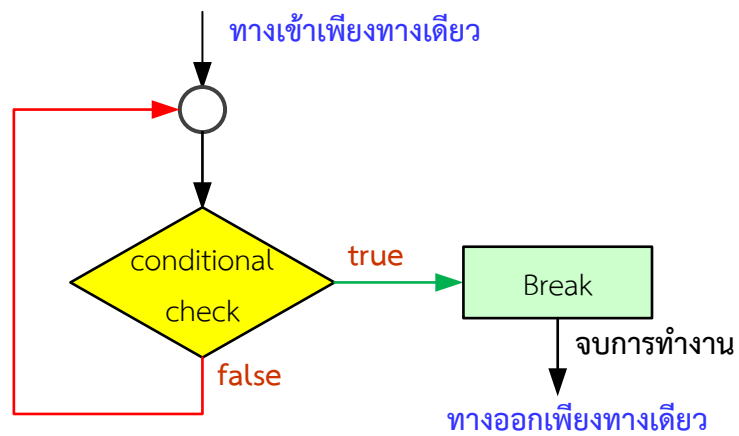
3.3.3. คำสั่ง for



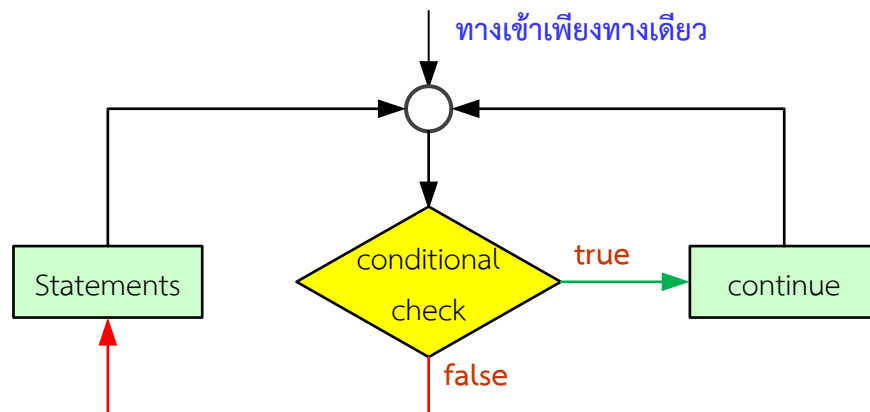


Flowchart ของคำสั่ง for

3.3.4. คำสั่ง break



3.3.5. คำสั่ง continue



3.3.6. คำสั่ง foreach...in

คำสั่ง foreach...in ใช้สำหรับการวนลูปเพื่อดึงค่าสมาชิกในอาเรย์ออกมาใช้ครั้งละ 1 ตัว โดยชนิดข้อมูลที่ดึงออกมาจะตั้งเป็นชนิดเดียวกับสมาชิกในอาเรย์

รูปแบบของคำสั่ง foreach...in
<pre>foreach (var_type var in var_array) { statements; }</pre>

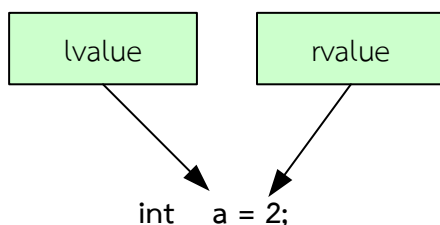
ตัวอย่างการใช้งานคำสั่ง foreach...in
<pre>class Program { static void Main(string[] args) { int[] collection = { 1, 2, 3, 4, 5 }; foreach (var item in collection) { Console.WriteLine(item); } } }</pre>

4. ตัวดำเนินการ (Operators)

ตัวดำเนินการในภาษา C# ส่วนใหญ่จะเหมือนกับตัวดำเนินการในภาษา C และ C++ แต่จะมีบางตัวที่เพิ่มความรัดกุมในการตรวจสอบขณะใช้งาน เช่น เครื่องหมาย = ซึ่งใช้ในการกำหนดค่าและ == ที่ใช้ในการเปรียบเทียบ

4.1. ตัวดำเนินการกำหนดค่า (Assignment Operators)

ตัวดำเนินการกำหนดค่า มีหน้าที่กำหนด rvalue ให้กับ lvalue



4.2. ตัวดำเนินการทางคณิตศาสตร์ (Mathematical Operators)

ภาษา C# มีตัวดำเนินการทางคณิตศาสตร์ทั้งหมด 5 ตัว ประกอบด้วยการกระทำทางคณิตศาสตร์ 4 ตัว ได้แก่การบวก (+) ลบ (-) คูณ (*)หาร (/) และตัวดำเนินการพิเศษที่ใช้หาค่าเศษจากการหารอีก 1 ตัว เรียกว่า modulo ใช้เครื่องหมาย % ในการดำเนินการ

4.3. ตัวดำเนินการเพิ่มค่า/ลดค่า ครั้งละ 1 หน่วย (Incremental/Decremental)

ตัวดำเนินการเพิ่มค่า/ลดค่าในภาษา C# มีลักษณะการใช้งานเช่นเดียวกับภาษา C นั่นคือใช้เครื่องหมาย ++ สำหรับการเพิ่มค่า และเครื่องหมาย -- สำหรับการลดค่าที่เก็บในตัวแปรขึ้นหรือลงครั้งละ 1 หน่วย ตัวอย่างเช่น

```
int a = 10;  
a++;
```

หลังจากดำเนินการแล้วจะได้ a = 11

4.4. ตัวดำเนินการเพิ่ม/ลดค่า (Calculate & reassign operator)

ตัวดำเนินการนี้จะต่างจาก Incremental/Decremental เนื่องจากเราสามารถระบุค่าที่ต้องการปรับปรุงลงไปในการดำเนินการ และสามารถใส่ตัวดำเนินการทางคณิตศาสตร์และตรรกะได้ด้วย ตัวดำเนินการนี้จะทำการคำนวณแล้วนำค่าที่ได้ใส่กลับลงไปในตัวแปรเดิม ดังตัวอย่าง

```
1 double mySalary = 1000d;
2 mySalary = mySalary * 1.10d;
```

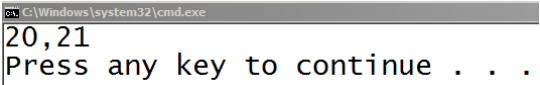
เราสามารถเขียนบรรทัดที่ 2 ได้เป็น

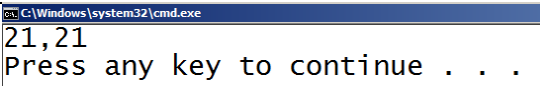
```
2 mySalary *= 1.10d;
```

นอกจากนี้เราสามารถใช้ตัวดำเนินการต่างๆ มาประกอบเป็น Calculate & reassign operator ได้ เช่น
+=, -=, *=, /=, %=

4.5. Prefix/Postfix operator

ตัวดำเนินการ Incremental/Decremental สามารถวางไว้ได้ทั้งด้านหน้าและด้านหลังตัวแปร ซึ่งลักษณะการวางทั้งสองตำแหน่งจะให้ผลในการดำเนินการที่แตกต่างกัน ดังตัวอย่าง

ตัวอย่าง Prefix:	
	<pre>int firstValue = 10; int secondValue = 20; firstValue = secondValue++; Console.WriteLine("{0},{1}", firstValue, secondValue);</pre>
Output:	
	

ตัวอย่าง Postfix:	
	<pre>int firstValue = 10; int secondValue = 20; firstValue = ++secondValue; Console.WriteLine("{0},{1}", firstValue, secondValue);</pre>
Output:	
	

4.6. ตัวดำเนินการเปรียบเทียบ (Relational Operators)

ตัวดำเนินการเปรียบเทียบ ใช้เปรียบเทียบค่า 2 ค่า อาจจะเป็นตัวแปรหรือค่าคงที่ต่างๆ ก็ได้ แล้วให้ผลการเปรียบเทียบเป็นชนิด boolean (true หรือ false) เท่านั้น ซึ่งผลจากการเปรียบเทียบอาจนำไปใช้กับคำสั่งเปลี่ยนทิศทางการดำเนินงานของโปรแกรม

ตัวอย่าง เมื่อกำหนดให้ bigValue = 100 และ smallValue = 50

ชื่อ	ตัวดำเนินการ	ตัวอย่าง	ผลที่ได้
Equals	==	bigValue == 100 bigValue == 80	true false
Not equals	!=	bigValue != 100 bigValue != 80	false true
Greater than	>	bigValue > smallValue	true
Greater than or equals	>=	bigValue >= smallValue smallValue >= bigValue	true false
Less than	<	bigValue < smallValue	false
Less than or equals	<=	smallValue <= bigValue bigValue <= smallValue	true false

4.7. Operator Precedence

ประเภท (ตามลำดับ ความสำคัญ)	ตัวดำเนินการ	ลำดับการ กระทำ
Primary	x.y f(x) a[x] x++ x-- new typeof default checked unchecked delegate	left
Unary	+ - ! ~ ++x --x (T)x	right
Multiplicative	* / %	left
Additive	+ -	left
Shift	<< >>	left
Relational	< > <= >= is as	left
Equality	== !=	right
Logical AND	&	left
Logical XOR	^	left
Logical OR		left
Conditional AND	&&	left
Conditional OR		left
Null Coalescing	??	left
Ternary	?:	right

Assignment	= *= /= %= += -= <<= >>= &= ^= = =>	right
------------	--------------------------------------	-------

4.8. The ternary operator

ตัวดำเนินการพิเศษตัวหนึ่งในภาษา C# เรียกว่า ternary operator ประกอบด้วยเครื่องหมาย 2 ตัว คือ ? และ : โดยมีรูปแบบการใช้งานคือ

cond-expr ? expr1 : expr2

การทำงานของตัวดำเนินการนี้จะเริ่มจากการพิจารณา cond-expr หน้าเครื่องหมาย ? ซึ่งต้องให้ผลลัพธ์เป็นชนิด boolean เท่านั้น ถ้าผลลัพธ์เป็น true จะทำงานในคำสั่ง expr1 ถ้าเป็น false จะทำงานในคำสั่ง expr2

5. อื่นๆ / Advanced features

ภาษา C# ยังมีลักษณะพิเศษอื่นๆ อีกมาก ซึ่งจะได้กล่าวถึงในโอกาสต่อไป