หลักภาษาเบสิก

1. บทนำ

จากบทที่ 3 ได้กล่าวถึงวิธีการใช้งาน mikroBasic PIC ในระดับเบื้องต้น คือ การสร้างโปรเจ็คต์ การเขียนโค้ด และการคอมไพล์ ซึ่งเมื่อได้ผลลัพธ์ที่เป็นแฟ้มฐานสิบหก ผู้พัฒนาต้องนำไปเขียนลงชิพด้วยโปรแกรม PICkit2 สุดท้ายจึงนำไปทดลองทำงานกับระบบที่ออกแบบไว้

ในบทนี้กล่าวถึงหลักของภาษาเบสิกของโปรแกรม mikroBasic PIC ที่ครอบคลุมถึงเรื่องที่ควรทราบ เพื่อใช้เป็นพื้นฐานของการเขียนโปรแกรมแบบโครงสร้างดังที่จะกล่าวถึงในบทที่ 5 ดังนั้น หากมีความรู้เกี่ยวกับภาษาเบสิกมาบ้างแล้ว จะสามารถอ่านหรือศึกษาบทนี้ได้อย่างรวดเร็วยิ่งขึ้น หรืออาจจะข้ามบทนี้ไปก่อนแล้วค่อยกลับมาศึกษาเพิ่มเติมเฉพาะส่วนที่ต้องการอ่านรายละเอียดเพิ่มเติม และสำหรับผู้เริ่มต้นฝึกเขียนภาษาเบสิกที่ต้องการศึกษาให้รวดเร็ว ควรใช้วิธีศึกษาจากตัวอย่างโปรแกรมในบทที่เกี่ยวกับการเชื่อมต่อหรือสั่งงานอุปกรณ์ไอ/โอที่ต้องการ และเมื่อมีข้อสงสัยจึงกลับมาศึกษาในเนื้อหาที่เกี่ยวข้องจะทำให้สามารถศึกษาการเขียนโปรแกรมสำหรับไมโครคอนโทรลเลอร์ PIC ได้รวดเร็วขึ้น

2. โครงสร้างหลักของโปรแกรม

โครงสร้างหลักของโปรแกรมภาษาเบสิกของ mikroBasic PIC เป็นดังตัวอย่างที่ 4.1

ตัวอย่างที่ 4.1 โครงสร้างหลักของโปรแกรมภาษาเบสิก

บรรทัด	โค้ด
1	program *ชื่อโปรแกรม*
2	' *ส่วนนี้สำหรับเขียนโค้ดโปรแกรม*
3	end.

โปรแกรมตัวอย่างที่ 4.1 เป็นโครงสร้างหลักของโปรแกรม ซึ่งประกอบไปด้วย 3 ส่วน คือ

1. ต้องมีการกำหนดชื่อโปรแกรม (บรรทัดที่ 2)
2. ส่วนของการเขียนโค้ดโปรแกรม
3. ส่วนจบการทำงานของโปรแกรม (บรรทัดที่ 3)

จากโครงสร้างหลักดังกล่าว ผู้เขียนโปรแกรมต้องเขียนโค้ดสั่งงานให้กับไมโครคอนโทรเลอร์ทำงานภายใต้ชุดคำสั่งที่ mikroBasic PIC เตรียมไว้ให้ หรือเขียนขึ้นมาเอง

3. การเขียนคำอธิบาย

การเขียนคำอธิบาย (Comment) คือ การเขียนข้อความเพื่อใช้ในการอธิบายสิ่งที่ทำ โดยข้อความเหล่านี้จะไม่ถูกนำมาแปลเป็นรหัสเครื่อง และถูกตัดทิ้งในช่วงของการแปลชุดคำสั่ง ดังนั้น ประโยชน์ของการเขียนคำอธิบาย คือ ใช้แทรกคำอธิบายการทำงานของโปรแกรม เพื่อผู้อื่นหรือผู้เขียนเองใช้สำหรับอ่านและทำความเข้าใจในสิ่งที่ได้เขียนเป็นโค้ดโปรแกรม

รูปแบบของการเขียนคำอธิบาย คือ

```
' ข้อความที่เป็นคำอธิบาย
```

จากรูปแบบของการเขียนคำอธิบายจะเห็นว่า อักขระที่อยู่ต่อจากเครื่องหมาย ' ตลอดทั้งบรรทัดถูกมองว่าเป็นคำอธิบายทั้งหมด ตัวอย่างการเขียนอธิบายเป็นดังตัวอย่างที่ 4.2

ตัวอย่างที่ 4.2 การเขียนคำอธิบาย

บรรทัด	โค้ด
1	' Project name : sample2
2	' Copyright (C), Jaroot Busarathid , 2010.
3	' MCU : PIC16F877
4	' Developer Board : CP-PIC/877 (ICD2)
5	' Oscillator : 10MHz
6	' Compiler : mikroBasic for PIC 2009 version 4.2
7	program sample2
8	end.

จากตัวอย่าง 4.2 พบว่า ตั้งแต่บรรทัดที่ 1 ถึง 6 เป็นการเขียนคำอธิบาย มีส่วนของการเขียนโปรแกรมมีเพียง 2 บรรทัด คือ บรรทัดที่ 7 และ 8 ซึ่งบรรทัดที่ 1 ถึง 6 ไม่ถูกนำมาแปลเป็นรหัสเครื่องและถูกตัดทิ้ง แต่การเขียนคำอธิบายทำให้ทราบว่า ผู้เขียนนั้นตั้งใจตั้งชื่อโปรเจ็คต์ว่า sample2 และเขียนโดย Jaroot Busarathid เมื่อปี 2010 ใช้ไมโครคอนโทรลเลอร์ยูนิต (MCU: Microcontroller Unit) รุ่น PIC16F877 ใช้บอร์ดสำหรับการพัฒนารุ่น CP-PIC/877 (ICD2) ใช้ความถี่ของสัญญาณนาฬิกาอยู่ที่ 10MHz และเขียนโปรแกรมด้วย mikroBasic for PIC 2009 version 4.2

4. สัญพจน์

สัญพจน์ (Literals) คือ ส่วนของข้อความหรือโทเคน (Token) ที่แสดงให้ทราบว่าเป็นค่าตัวเลข (Numeric Literal) ทั้งที่เป็นสัญพจน์เลขจำนวนเต็ม (Integer Literals) หรือสัญพจน์เลขทศนิยม (Floating Point Literals), ค่าของตัวอักษร (Character Literal) หรือค่าของข้อความ (String Literal) ดังรูปแบบต่อไปนี้

1. สัญพจน์เลขจำนวนเต็ม คือ การเขียนอักขระที่สื่อถึงตัวเลขจำนวนเต็ม เขียนได้ 3 รูปแบบ คือ เลขฐานสิบ (Decimal) เลขฐานสิบหก (Hexadecimal) และเลขฐานสอง (Binary) ดังนี้

- เลขฐานสิบใช้อักษระ 0 1 2 3 4 5 6 7 8 และ 9 แทนค่าในแต่ละหลัก โดยใช้เครื่องหมาย + และ – นำหน้าตัวเลขเพื่อระบุว่าเป็นจำนวนบวก หรือจำนวนลบ
- เลขฐานสิบหกใช้อักขระ 0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E และ F แทนค่าแต่ละหลัก โดยต้องนำหน้าตัวเลขด้วยเครื่องหมาย $ หรือ 0x (ศูนย์เอ็กซ์) เพื่อเป็นการบอกว่าตัวเลขนั้นเป็นเลขฐานสิบหก โดยการเทียบค่าของอักขระ a ถึง f และ A ถึง F เป็นตัวเลขฐานสิบนั้นสามารถเทียบได้ดังตารางที่ 4.1

ตารางที่ 4.1 ตารางเปรียบเทียบอักขระของเลขฐานสิบหกกับเลขฐานสิบ

อักขระฐานสิบหก		ค่าในฐานสิบ
A	a	10
B	b	11
C	c	12
D	d	13
E	e	14
F	f	15

- เลขฐานสองจะประกอบไปด้วยอักขระ 0 หรือ 1 เป็นค่าในแต่ละหลัก โดยด้านหน้าตัวเลขต้องขึ้นต้นด้วยเครื่องหมาย % เพื่อเป็นการบอกว่าตัวเลขเหล่านั้นเป็นเลขฐานสอง

2. สัญพจน์เลขทศนิยม คือ ตัวเลขที่เขียนแทนด้วย 3 ส่วน คือ

- ตัวเลขจำนวนเต็ม (Decimal Integer) ต้องเขียนด้วยเลขฐานสิบ
- เครื่องหมาย . (Decimal Point) สำหรับคั่นเลขจำนวนเต็มกับเศษ
- ตัวเลขเศษ (Decimal Fraction) ต้องเขียนด้วยเลขฐานสิบ

นอกจากนี้ ผู้เขียนโปรแกรมอาจจะใช้สัญลักษณ์ e หรือ E เพื่อใช้เป็นตัวบอกค่ากำลังสิบ เช่น 4e8 จะหมายถึง 4×10^{8} หรือ $1.8e^{-2}$ แทน 1.8×10^{-2} เป็นต้น

3. สัญพจน์อักขระ คือ อักขระจำนวน 1 อักษร ซึ่งอยู่ในรูปแบบของตาราง ASCII ซึ่งต้องคร่อมด้วยเครื่องหมาย " เสมอ (ในเครื่องหมายนี้ต้องมีอักขระเพียง 1 ตัวเท่านั้น)

4. สัญพจน์สายอักขระ คือ อักขระหลาย ๆ ตัวเรียงต่อกันได้สูงสุด 255 อักษร และต้องคร่อมด้วยเครื่องหมาย " เช่นเดียวกับอักขระ

ตัวอย่างที่ 4.3 การใช้สัญพจน์

บรรทัด	โค้ด
1	program sample3
2	const int_dec = 128
3	const int_hex1 = 0x128
4	const int_hex2 = $128
5	const int_bin = %10000000
6	const float_1 = 128.0
7	const float_2 = 1.28e2
8	const ch_val = "1"
9	const st_val = "128"
10	end.

จากตัวอย่าง 4.3 เป็นการกำหนดค่าคงที่ 6 ค่า คือ int_dec int_hex1 int_hex2 int_bin ch_val และ st_val โดยค่าคงที่แต่ละตัวมีค่าดังนี้

int_dec มีค่าเป็น 128 ในเลขฐานสิบ

int_hex1 และ int_hex2 มีค่าเป็น 296 ในเลขฐาน 10

int_bin มีค่าเป็น 128 ในเลขฐานสิบ

float_1 มีค่าเป็น 128.0

float_2 มีค่าเป็น 1.28e2 หรือ 1.28 x10^2 หรือ 128.0

ch_val มีค่าเป็นอักขระ 1

st_val มีค่าเป็นสายอักขระหรือข้อความ 128

5. คำสำคัญ

คำสำคัญ (Key Word) เป็นคำที่ถูกใช้โดยเป็นชุดคำสั่งภาษาเบสิก และเป็นคำที่ผู้เขียนโปรแกรมไม่สามารถใช้เป็นชื่อของตัวแปร หรือฟังก์ชันได้ คำที่ mikroBasic PIC ถือว่าเป็นคำสำคัญมีดังนี้

abs	csng	for	line	private	step
and	cstr	form	loc	property	stop
appactivate	curdir	format lock	pset	str	
array	currency	forward	lof	public	
strcomp					
as	cvar	freefile	log	put	strconv
asc	cverr	fv	long	pv	string
asm	date	get	longint	qbcolor	sub
atn	dateadd	getattr	loop	raise	switch
attribute	datediff	getobject	lset	randomize	syd
base	datepart	gosub	ltrin	rate	system
bdata	dateserial	goto	me	real	tab
beep	datevalue	hex	mid	redim	tan
begin	ddb	hour	minute	rem	time
bit	deftype	idata	mirr	reset	timer
boolean	dim	if	mkdir	resume	
timeserial					
byte	dir	iif	mod	return	
timevalue					

call	div	ilevel	module	rgb	to
case	do	imp	month	right	trim
cbool	doevents	include	msgbox	rmdir	typedef
cbyte	double	input	name	rnd	
typename					
ccur	each	instr	new	rset	ubound
cdate	empty	int	next	rtrim	ucase
	end	integer	not	sbit	unlock
cdbl		ipmt		second	util
char	environ	irr	nothing	seek	val
chdir	eof	is	now	select	variant
chdrive	eqv	isarray	nper	sendkeys	vartype
chr	erase	isdate	npv	set	version
cint	err	isempty	object	setattr	volatile
circle	error	iserror	oct	sfr	
weekday					
class	exit	ismissing	on	sgn	wend
clear	exp	isnull	open	shell	while
clng	explicit	isnumeric	option	short	width
close		isobject		sin	with
code	fileattr	kill		single	word
command		large	or	sln	write
compact	filecopy	lbound	org	small	xdata
compare	filedatetime	lcase	pdata	space	xor
const	filelen	left	pmt	spc	
cos	fix	len	ppmt	sqr	
createobject	float	let	print	static	

6. การตั้งชื่อ

การตั้งชื่อ หรือตัวระบุ (Identifier) คือ การใช้อักขระแทนชื่อสิ่งใดสิ่งหนึ่ง รูปแบบของการตั้งชื่อเป็นดังนี้

1. ขึ้นต้นด้วยอักขระ a ถึง z หรือ A ถึง Z หรือเครื่องหมาย _
2. อักขระตัวต่อไปอาจจะเป็นตามข้อ 1 หรือเป็น 0 ถึง 9
3. ในชื่อเดียวกันจะต้องเป็นตามข้อ 1 หรือ 2 อยู่เรียงต่อกันไป
4. ตัวอักขระพิมพ์เล็ก (a ถึง z) หรือพิมพ์ใหญ่ (A ถึง Z) ถือว่าเป็นอักขระเดียวกัน
5. ต้องไม่ซ้ำกับคำสำคัญ

จากข้อกำหนดทั้ง 5 ข้อ สามารถนำมาเขียนเป็นชื่อหรือตัวระบุที่ถูกต้องตามรูปแบบและไม่ถูกต้องตามรูปแบบดังตารางที่ 4.2

ตารางที่4.2 ตัวอย่างการตั้งชื่อ

ชื่อ	ถูก/ผิด (เหตุผล)
TheMan	ถูกต้อง
THEMan	ถูกต้อง แต่ถือว่าเป็นคำเดียวกับ TheMan
The Man	ผิด เพราะมีช่องว่างคั่นระหว่าง The กับ Man
__HeyMan	ถูกต้อง
_Data_From_USB	ถูกต้อง
1st_Number	ผิด เนื่องจากขึ้นต้นด้วยตัวเลข 1
The_1st_Number	ถูกต้อง
$MyHex	ผิด เพราะขึ้นต้นด้วยเครื่องหมาย $
%101101	ผิด เพราะเป็นการกำหนดค่าตัวเลขฐานสอง
x123ABF	ถูกต้อง
The%_of_VAT	ผิด เพราะใช้เครื่องหมาย % ในชื่อ
Score_Number_1	ถูกต้อง
IsArray	ผิด เพราะซ้ำกับคำสำคัญ
Is_Array	ถูกต้อง

7. ประเภทของตัวแปร

ในระบบไมโครคอนโทรลเลอร์จัดเก็บข้อมูลตามประเภทของข้อมูลที่กำหนดโดยผู้เขียนโปรแกรม ซึ่งตัวแปรแต่ละประเภทมีขนาดและช่วงค่าที่แตกต่างกัน ทั้งนี้เพื่อให้เลือกใช้ตามความเหมาะสม ในตัวแปลภาษา mikroBasic PIC ได้กำหนดตัวแปรพื้นฐานแบ่งเป็นประเภทต่างๆ ดังตารางที่ 4.3

ตารางที่ 4.3 ประเภทของตัวแปร

ประเภทตัวแปร	ขนาด (บิต)	ช่วงค่า
Byte	8	0..255
Char	8	ASCII 0..255
String	ตัวละ 8	ข้อความที่ประกอบไปด้วย ASCII 0..255
Word	16	0..65535
Short	8	-128..127
Integer	16	-32768..32757
Longint	32	-2147483648..2147483647
Float	32	+/-1.17549435082E-38..+/-6.80564774407E38

นอกจากประเภทตัวแปรพื้นฐานทั้ง 8 แล้ว ยังมีตัวแปรพิเศษที่จะกล่าวถึงในหัวข้อถัดไปอีก คือ ตัวแปรแบบแถวลำดับหรืออะเรย์ (Array) และตัวแปรตัวชี้หรือพอยน์เตอร์ (Pointer) ซึ่งเป็นชนิดตัวแปรที่นำไปประยุกต์ใช้ได้หลากหลาย เช่น ใช้ในการจัดการกับข้อมูลแบบกลุ่ม ใช้ในการชี้ตัวแปรอื่นๆ ในหน่วยความจำ เพื่อเข้าไปแก้ไขค่า หรือนำค่ามาใช้ เป็นต้น

8. ตัวแปร

ตัวแปร (Variable) คือ การกำหนดหน่วยความจำเพื่อใช้สำหรับจัดเก็บข้อมูล และใช้ตัวระบุเป็นชื่อในการอ้างอิงแทนการอ้างอิงเป็นตำแหน่งของตัวแปรในหน่วยความจำ โดยข้อมูลในตัวแปรเปลี่ยนแปลงได้ตลอดการการทำงานของโปรแกรมทำงานนั้น รูปแบบของการกำหนดตัวแปรเป็นดังนี้

dim ชื่อตัวแปร **as** ชนิดตัวแปร

ในกรณีของการประกาศตัวแปรชนิดเดียวกันพร้อมกันหลายตัวโดยไม่ต้องแบ่งการประกาศตัวแปรเฉพาะแต่ละตัว ให้ใช้เครื่องหมาย , คั่นระหว่างชื่อของแต่ละตัวแปร ดังรูปแบบต่อไปนี้

dim ชื่อตัวแปร$_1$, ชื่อตัวแปร$_2$, ..., ชื่อตัวแปร$_N$ **as** ชนิดตัวแปร

ตัวอย่างที่ 4.4 การประกาศตัวแปร

บรรทัด	โค้ด
1	program sample4
2	dim sensor1 as integer
3	dim sw_left as Boolean
4	dim sw_right as Boolean
5	dim counter as integer
6	dim average as float
7	dim ascii as byte
8	end.

จากตัวอย่างที่ 4.4 จะเห็นว่ามีการประกาศตัวแปรทั้งสิ้น 6 ตัวแปร ดังรายละเอียดต่อไปนี้

sensor1 เป็นตัวแปรชนิด integer

sw_right เป็นตัวแปรชนิด boolean

sw_left เป็นตัวแปรชนิด boolean

counter เป็นตัวแปรชนิด integer

average เป็นตัวแปรชนิด float

ascii เป็นตัวแปรชนิด byte

9. ค่าคงที่

ค่าคงที่ (Constant) คือ การกำหนดชื่อเรียกของค่าใดค่าหนึ่ง อันเป็นได้ทั้ง จำนวนเต็ม จำนวนทศนิยม และตัวอักษร ซึ่งค่าของชื่อที่กำหนดขึ้นจะไม่เปลี่ยนแปลงในขณะที่โปรแกรมทำงาน รูปแบบของการกำหนดค่าคงที่เป็นดังนี้

const ชื่อค่าคงที่ [**as** ชนิดค่าคงที่] = ค่าที่กำหนด

หมายเหตุ

ชนิดค่าคงที่ คือ ชนิดของตัวแปร

ตัวอย่างที่ 4.5 การประกาศค่าคงที่

บรรทัด	โค้ด
1	program sample5
2	const MAX as longint = 10000
3	const MIN = 100
4	const bSWITCH = "n"
5	const MSG = "Hello"
6	const PRICE1 = 25.75
7	const VOLTAGE as float = 4.5
8	const myPI = 4.14159
9	end.

จากตัวอย่างที่ 4.5 จะเห็นว่ามีการประกาศชื่อของค่าคงที่ทั้งสิ้น 7 ตัว ดังรายละเอียดต่อไปนี้

MAX เป็นค่าคงที่ชนิด longint มีค่าเป็น 10000

MIN เป็นค่าคงที่มีค่าเป็น 100

bSWITCH เป็นค่าคงที่มีค่าเป็นอักขระ n

MSG เป็นค่าคงที่มีค่าเป็นข้อความว่า Hello

VOLTAGE เป็นค่าคงที่ชนิด float มีค่าเป็น 4.5

myPI เป็นค่าคงที่มีค่าเป็น 4.14159

10. เลเบล

เลเบล (Label) คือ ชื่อเรียกที่ถูกกำหนดขึ้นมาโดยผู้เขียนโปรแกรม เพื่อใช้สำหรับเป็นตำแหน่งอ้างอิงสำหรับคำสั่ง goto หรือ gosub โดยรูปแบบของการกำหนดเลเบลของภาษาเบสิกเป็นดังนี้

ชื่อเลเบล:

ตัวอย่างที่ 4.6 การประกาศค่าคงที่

บรรทัด	โค้ด
1	program sample6
2	main:
3	dim bStatus as Boolean
4	bStatus = true
5	loop1:
6	bStatus = not bStatus
7	goto loop1
8	end.

จากตัวอย่าง 4.6 เป็นตัวอย่างการสร้างเลเบล มีรายละเอียดรายบรรทัดดังนี้

บรรทัดที่ 2 ประกาศเลเบลที่ชื่อว่า main

บรรทัดที่ 3 ประกาศตัวแปรชื่อ bStatus ให้เป็นตัวแปรชนิด Boolean

บรรทัดที่ 4 กำหนดให้ตัวแปร bStatus มีค่าเป็น true

บรรทัดที่ 5 ประกาศเลเบลที่ชื่อว่า loop1

บรรทัดที่ 6 กำหนดให้ตัวแปร bStatus มีค่าเป็นส่วนกลับของค่าที่เก็บในตัวแปร bStatus

บรรทัดที่ 7 กระโดดไปทำถ้อยแถลงที่อยู่ต่อเลเบล loop1

11. สัญลักษณ์

สัญลักษณ์ (Symbol) คือ การใช้ชื่อเรียกแทนบางสิ่งหรือการทำแมโคร (Macro) เพื่อใช้แทนบางสิ่ง เช่น ถ้าต้องการสั่งให้มีการหน่วงเวลาเป็นระยะเวลา 500 มิลลิวินาที จะต้องใช้คำสั่งว่า delay_ms(500) แต่เพื่อให้สะดวกต่อการพิมพ์โค้ด ผู้เขียนโปรแกรมอาจจะใช้สัญลักษณ์หรือสร้างแมโครเพื่อใช้เรียกแทนคำว่า delay_ms(500) ทำให้การพิมพ์คำสั่งสั้นลง หรือถ้าในโปรแกรมมีการเรียกคำสั่ง delay_ms(500) หลายจุด แล้วต้องการเปลี่ยนแปลงให้มีการหน่วงเวลาที่เปลี่ยนไปจากเดิม ผู้เขียนโปรแกรมจะต้องแก้ไขชุดคำสั่งทุกคำสั่ง ซึ่งถ้าใช้แมโคร จะสะดวกกว่ามาก นั่นคือผู้เขียนโปรแกรมเพียงเปลี่ยนแปลงที่แมโครจะทำให้ทุกที่ที่เรียกแมโครนั้นเปลี่ยนตามไปด้วย รูปแบบของการสร้างสัญลักษณ์เป็นดังนี้

symbol คำเรียก = สิ่งที่ใช้แทน

หมายเหตุ

ความแตกต่างของสัญลักษณ์กับค่าคงที่ คือ สัญลักษณ์เป็นการใช้ชื่อเรียกแทนบางสิ่ง ซึ่งคำเรียกแทนนี้จะถูกตัวแปลภาษานำค่าที่ใช้แทนมาใส่ในตอนแปลภาษา ส่วนค่าคงที่เป็นการจองหน่วยความจำของโปรแกรมเพื่อใช้เก็บค่าที่ผู้เขียนโปรแกรมระบุ

ตัวอย่างที่ 4.7 การสร้างสัญลักษณ์หรือแมโคร

บรรทัด	โค้ด
1	program sample7
2	symbol MaxLoop = 1000
3	symbol MyDelay = Delay_ms(500)
4	dim MyCounter as Longint
5	main:
6	MyCounter = 0
7	loop1:
8	MyCounter = MyCounter + 1
9	MyDelay
10	if MyCounter < MaxLoop then
11	goto loop1
12	end if
13	end.

จากตัวอย่าง 4.7 มีการสร้างสัญลักษณ์ขึ้นมา 2 สัญลักษณ์ คือ MaxLoop และ MyDelay ซึ่งมีความหมายดังนี้

MaxLoop ใช้แทนค่าตัวเลขจำนวนเต็มที่มีค่าเป็น 1000

MyDelay ใช้แทนชุดคำสั่ง Delay_ms(500)

ด้วยเหตุนี้ บรรทัดที่ 9 ที่มีการเรียกคำสั่ง MyDelay จึงมีความหมายเดียวกันกับการเรียกคำสั่ง Delay_ms(500) และบรรทัดที่ 10 จึงมีความหมายเดียวกันกับการพิมพ์ชุดคำสั่งว่า if MyCounter < 1000 then แต่อย่างไรก็ดี การสร้างเป็นสัญลักษณ์จะอำนวยความสะดวกในการปรับแก้โปรแกรมมากกว่าการใช้คำสั่งโดยตรง นั่นคือ ถ้าเปลี่ยนคำสั่งบรรทัดที่ 2 เป็นคำสั่งด้านล่างจะทำให้ผู้เขียนโปรแกรมไม่ต้องเปลี่ยนค่าใดๆ ในบรรทัดที่ 10 เนื่องจาก ณ ทุกที่ที่ใช้สัญลักษณ์เป็นคำว่า MaxLoop จะมีค่าเท่ากับที่กำหนดเสมอ นั่นก็คือ จะมีค่าเป็น 2000 ไปโดยปริยาย

```
symbol  MaxLoop = 2000
```

และเช่นเดียวกัน ถ้าต้องการเปลี่ยนการหน่วงเวลาให้เป็น 2000 มิลลิวินาที ผู้เขียนโปรแกรมสามารถเปลี่ยนแปลงคำสั่งของบรรทัดที่ 3 ให้เป็นดังคำสั่งด้านล่าง แล้วจะทำให้คำสั่งบรรทัดที่ 9 มีความหมายเป็น Delay_ms(2000) ไปด้วย

```
symbol MyDelay = Delay_ms(2000)
```

ดังนั้น เมื่อโปรแกรม 4.7 ถูกแปลภาษาด้วย mikroBasic PIC จะได้ผลลัพธ์ของโค้ดโปรแกรมเป็นดังนี้

```
program sample07
dim  MyCounter as Longint
main:
     MyCounter = 0
loop1:
    MyCounter = MyCounter + 1
    Delay_ms(500)
    if MyCounter < 1000 then
       goto loop1
    end if
end.
```


12. ตัวแปรแถวลำดับ

ตัวแปรแถวลำดับ (Array Variable) คือ โครงสร้างข้อมูลพื้นฐานประเภทหนึ่งที่มีคุณสมบัติดังต่อไปนี้

1. มีการกำหนดจำนวนสมาชิกที่แน่นอน
2. สมาชิกแต่ละตัวเป็นตัวแปรประเภทเดียวกัน
3. สมาชิกแต่ละตัวจะต้องอยู่เรียงกันจากสมาชิกลำดับแรกไปถึงสมาชิกลำดับสุดท้าย

ข้อดีของการใช้แถวลำดับคือ ทำให้ผู้เขียนโปรแกรมสามารถจัดหมวดหมู่ของข้อมูลให้เป็นกลุ่มก้อน และใช้เทคนิคการเขียนโปรแกรมแบบวนรอบเพื่อรับข้อมูล หรือประมวลผลข้อมูล หรือนำข้อมูลแสดงผลได้ง่าย เช่น ถ้าต้องรับค่าแรงดันจากภายนอกมาจำนวน 10 ชุด หลังจากนั้นคำนวณหาค่าเฉลี่ยของแรงดันทั้งหมด และนำผลส่งออกไปที่พอร์ต C นั้น ถ้าเป็นการเขียนโปรแกรมโดยทั่วไปต้องมีการประกาศตัวแปรเพื่อเก็บข้อมูลทั้ง 10 ชุด ดังนี้

```
Dim Data1, Data2, Data3, Data4, Data5 as Integer
Dim Data6, Data7, Data8, Data9, Data10 as Integer
```

ในการรับค่าแรงดันจาก AN1 ด้วยคำสั่ง Adc_Read(1) สามารถเขียนเป็นโค้ดได้ดังต่อไปนี้

```
Data1 = Adc_Read(1)
Data2 = Adc_Read(1)
Data3 = Adc_Read(1)
Data4 = Adc_Read(1)
Data5 = Adc_Read(1)
Data6 = Adc_Read(1)
Data7 = Adc_Read(1)
Data8 = Adc_Read(1)
Data9 = Adc_Read(1)
Data10 = Adc_Read(1)
```

จากที่กล่าวมานี้ ถ้ามีการเปลี่ยนแปลงข้อกำหนดให้รับค่าแรงดันจำนวน 20 ค่า จะทำให้ผู้เขียนโปรแกรมต้องแก้ไขโค้ดโปรแกรมหลายจุด นั่นคือ ต้องจองตัวแปร (Allocate) จาก 10 ตัวแปร (Data1 ถึง Data10) มาเป็น 20 ตัวแปร (เพิ่ม Data11 ถึง Data20) ดังนี้

```
Dim Data1, Data2, Data3, Data4, Data5 as Integer
Dim Data6, Data7, Data8, Data9, Data10 as Integer
Dim Data11, Data12, Data13, Data14, Data15 as Integer
Dim Data16, Data17, Data18, Data19, Data20 as Integer
```

และต้องเขียนคำสั่งรับค่าจากเดิม 10 คำสั่ง มาเป็น 20 คำสั่ง คือ

```
Data1 = Adc_Read(1)
Data2 = Adc_Read(1)
Data3 = Adc_Read(1)
Data4 = Adc_Read(1)
Data5 = Adc_Read(1)
…
Data19 = Adc_Read(1)
Data20 = Adc_Read(1)
```

แต่ถ้าเปลี่ยนชนิดตัวแปรมาเป็นตัวแปรแถวลำดับจะทำให้การประกาศตัวแปรเปลี่ยนไป คือ ผู้เขียนโปรแกรมสามารถใช้ชื่อตัวแปรเป็นชื่อเดียวกัน และต้องกำหนดจำนวนสมาชิกที่ต้องการใช้ ทำให้ได้ชุดคำสั่งสำหรับการประกาศตัวแปรสำหรับรับค่าแรงดัน 10 ข้อมูลดังนี้

```
Dim  DataN as integer[10]
```

และเขียนคำสั่งสำหรับการรับค่าด้วยการวนรอบได้ดังนี้

```
for  cnt=1 to 10
     DataN[cnt] = Adc_Read(1)
next  cnt
```


เมื่อต้องการเปลี่ยนจำนวนข้อมูลจาก 10 มาเป็น 20 ข้อมูลจะเปลี่ยนเฉพาะจำนวนสมาชิก และจำนวนครั้งของการวนรอบ ดังโค้ดด้านล่างนี้

```
Dim  DataN as integer[20]
…
for  cnt=0 to 19
     DataN[cnt] = Adc_Read(1)
next  cnt
```

จากตัวอย่างจะเห็นว่าการที่ผู้เขียนโปรแกรมรู้จักการใช้ตัวแปรแบบแถวลำดับจะช่วยให้การเขียนโปรแกรมยืดหยุ่นขึ้น และสะดวกต่อการปรับปรุงแก้ไขโปรแกรมได้ดีกว่าเขียนโปรแกรมโดยไม่ได้ใช้ตัวแปรแบบแถวลำดับ

รูปแบบการสร้างตัวแปรแถวลำดับ เป็นดังนี้

dim ชื่อตัวแปร as ชนิดตัวแปร[จำนวนสมาชิก]

รูปแบบการสร้างตัวแปรแถวลำดับ 2 มิติ เป็นดังนี้

dim ชื่อตัวแปร as ชนิดตัวแปร[จำนวนแถว][จำนวนสมาชิกแต่ละแถว]

สำหรับการนำค่าป้อนให้กับตัวแปรแถวลำดับมีรูปแบบดังนี้

ชื่อตัวแปร[ลำดับสมาชิก] = ค่านำเข้า

กรณีของการนำค่าป้อนให้กับตัวแปรแถวลำดับ 2 มิติมีรูปแบบดังต่อไปนี้

ชื่อตัวแปร[ลำดับแถวของสมาชิก][ลำดับสมาชิกในแถว] = ค่านำเข้า

รูปแบบของการอ้างอิงค่าจากตัวแปรแถวลำดับ คือ

ชื่อตัวแปร[ลำดับสมาชิก]

และรูปแบบของการอ้างอิงค่าจากตัวแปรแถวลำดับ 2 มิติ คือ

ชื่อตัวแปร[ลำดับแถวของสมาชิก][ลำดับสมาชิกในแถว]

หมายเหตุ

ลำดับเริ่มต้นของแถวลำดับ คือ 0 ดังนั้น ถ้ากำหนดจำนวนสมาชิกเป็น 20 จะสามารถอ้างอิงค่าลำดับได้ตั้งแต่ 0 ถึง 19 และเช่นเดียวกันกับการใช้กับแถวลำดับ 2 มิติ ค่าลำดับของแถวจะเริ่มจาก 0 และลำดับของสมาชิกตัวแรกในแถวจะเริ่มจาก 0 ด้วยเช่นกัน

ตัวอย่างที่ 4.8 การใช้แถวลำดับ

บรรทัด	โค้ด
1	program sample8
2	symbol MaxDataN = 10
3	dim dataN as integer[MaxDataN]
4	dim cnt as integer
5	dim SumData as integer
6	dim AverageData as float
7	main:
8	' ว่างไว้สำหรับเพิ่มโค้ดกำหนดการทำงานของ AN1
9	SumData = 0
10	for cnt=0 to MaxDataN-1
11	dataN[cnt] = Adc_Read(1)
12	SumData = SumData + dataN[cnt]
13	next cnt
14	AverageData = SumData / MaxDataN
15	end.

จากตัวอย่าง 4.8 เห็นว่าเมื่อนำเรื่องของการใช้สัญลักษณ์หรือแมโครเพื่อเป็นคำเรียกแทนจำนวนสมาชิกยิ่งทำให้การเขียนโปรแกรมนั้นยืดหยุ่นมากยิ่งขึ้น เมื่อมีการเปลี่ยนจำนวนข้อมูลจาก 10 ชุดเป็น 20 ชุด ผู้เขียนโปรแกรมเพียงเปลี่ยนบรรทัดที่ 2 ให้เป็นดังคำสั่งด้านล่าง จะทำให้โปรแกรมรองรับการทำงานสำหรับข้อมูล 20 ชุดโดยปริยาย


```
symbol MaxDataN = 20
```

13. แถวลำดับของค่าคงที่

ความแตกต่างของแถวลำดับของค่าคงที่ (Constant Array) เมื่อเปรียบเทียบกับตัวแปรแถวลำดับลำดับ คือ แถวลำดับประเภทนี้ใช้สำหรับเก็บค่าที่เป็นค่าคงที่เท่านั้น ดังนั้น ค่าใดๆ ของสมาชิกในแถวลำดับประเภทนี้จะไม่สามารถถูกแก้ไขได้ ซึ่งรูปแบบของแถวลำดับของค่าคงที่เป็นดังนี้

symbol ชื่อค่าคงที่ **as** ชนิดของค่าคงที่[จำนวนสมาชิก]
= (ค่าที่1, ค่าที่2, ..., ค่าที่N)

รูปแบบการกำหนดแถวลำดับของค่าคงที่แบบ 2 มิติ คือ

symbol ชื่อค่าคงที่ **as** ชนิดของค่าคงที่[จำนวนแถว][จำนวนสมาชิกในแถว]
= ((ค่าที่1ของแถวที่ 1, ค่าที่2 ของแถวที่ 1, ..., ค่าที่N ของแถวที่ 1),
(ค่าที่1ของแถวที่ 1, ค่าที่2 ของแถวที่ 1, ..., ค่าที่N ของแถวที่ 1),
...
(ค่าที่1ของแถวที่ M, ค่าที่2 ของแถวที่ M, ..., ค่าที่N ของแถวที่ M),)

สำหรับรูปแบบของการอ้างอิงค่าจากแถวลำดับของค่าคงที่นั้นจะมีรูปแบบเช่นเดียวกันกับการอ้างอิงค่าจากตัวแปรแถวลำดับที่ได้กล่าวถึงมาก่อนหน้านี้

ตัวอย่างที่ 4.9 การใช้แถวลำดับของค่าคงที่

บรรทัด	โค้ด
1	program sample9
2	symbol x = 0
3	symbol y = 1
4	symbol z = 2
5	const P1 as integer[3] = (10, 10, 5)
6	const Matrix as integer[3][3] = (
7	(1,0,0),
8	(0,2,0),
9	(0,0,1)
10	)
11	dim P2 as integer[3]
12	main:
13	P2[x]=P1[x]*Matrix[0][0]+P1[y]*Matrix[1][0]+P1[z]*Matrix[2][0]
14	P2[y]=P1[x]*Matrix[0][1]+P1[y]*Matrix[1][1]+P1[z]*Matrix[2][1]
15	P2[z]=P1[x]*Matrix[0][2]+P1[y]*Matrix[1][2]+P1[z]*Matrix[2][2]
16	end.

ตัวอย่างโปรแกรม 4.9 เป็นตัวอย่างการคูณเมตริกขนาด 1x3 กับ 3x3 โดยใช้ค่าคงที่ P1 และ Matrix เก็บข้อมูล และเก็บผลของการคำนวณในตัวแปร P2 โดยขั้นตอนการทำงานเป็นดังนี้

บรรทัดที่ 2 สร้างสัญลักษณ์ชื่อว่า x ให้มีค่าเป็น 0

บรรทัดที่ 3 สร้างสัญลักษณ์ชื่อว่า y ให้มีค่าเป็น 1

บรรทัดที่ 4 สร้างสัญลักษณ์ชื่อว่า z ให้มีค่าเป็น 2

บรรทัดที่ 5 สร้างค่าคงที่ชื่อว่า P1 ให้เป็นแถวลำดับของค่าคงที่ชนิด integer จำนวน 3 สมาชิก และกำหนดค่าเป็น 10, 10 และ 5 ตามลำดับ

บรรทัดที่ 6-10 สร้างค่าคงที่ชื่อว่า Matrix ให้เป็นแถวลำดับของค่าคงที่ชนิด integer จำนวน 3 แถว แถวละ 3 สมาชิก (รวมเป็น 9 สมาชิก) โดยกำหนดค่าให้แถวแรก หรือแถวที่ 0 มีค่าเป็น 1, 0 และ 0 ตามลำดับ กำหนดให้แถวที่ 1 มีค่าเป็น 0, 2 และ 0 ตามลำดับ และกำหนดให้แถวสุดท้ายหรือแถวที่ 2 มีค่าเป็น 0, 0 และ 1 ตามลำดับ

บรรทัดที่ 11 ประกาศตัวแปรชื่อว่า P2 ให้เป็นตัวแปรชนิดแถวลำดับชนิด integer มี 3 สมาชิก

บรรทัดที่ 13 ให้ P2 ตัวที่ x เก็บผลลัพธ์จากการประมวลผลค่าตามนิพจน์ดังต่อไปนี้

```
P1[x]*Matrix[0][0]+P1[y]*Matrix[1][0]+P1[z]*
Matrix[2][0]
```


บรรทัดที่ 14 ให้ P2 ตัวที่ y เก็บผลลัพธ์จากการประมวลผลค่าตามนิพจน์ดังต่อไปนี้

```
P1[x]*Matrix[0][1]+P1[y]*Matrix[1][1]+P1[z]*
Matrix[2][1]
```

บรรทัดที่ 15 ให้ P2 ตัวที่ z เก็บผลลัพธ์จากการประมวลผลค่าตามนิพจน์ดังต่อไปนี้

```
P1[x]*Matrix[0][2]+P1[y]*Matrix[1][2]+P1[z]*
Matrix[2][2]
```

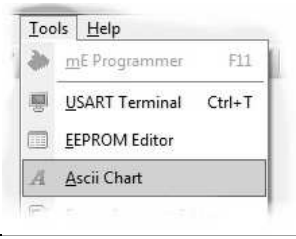
14. สายอักขระ

สายอักขระ (String) คือ ประเภทของข้อมูลชนิดหนึ่งที่ใช้สำหรับเก็บอักขระที่อยู่เรียงกันหรืออีกนัยหนึ่ง สายอักขระก็คือแถวลำดับของข้อมูลแบบอักขระ รูปแบบของการสร้างตัวแปรสายอักขระเป็นดังนี้

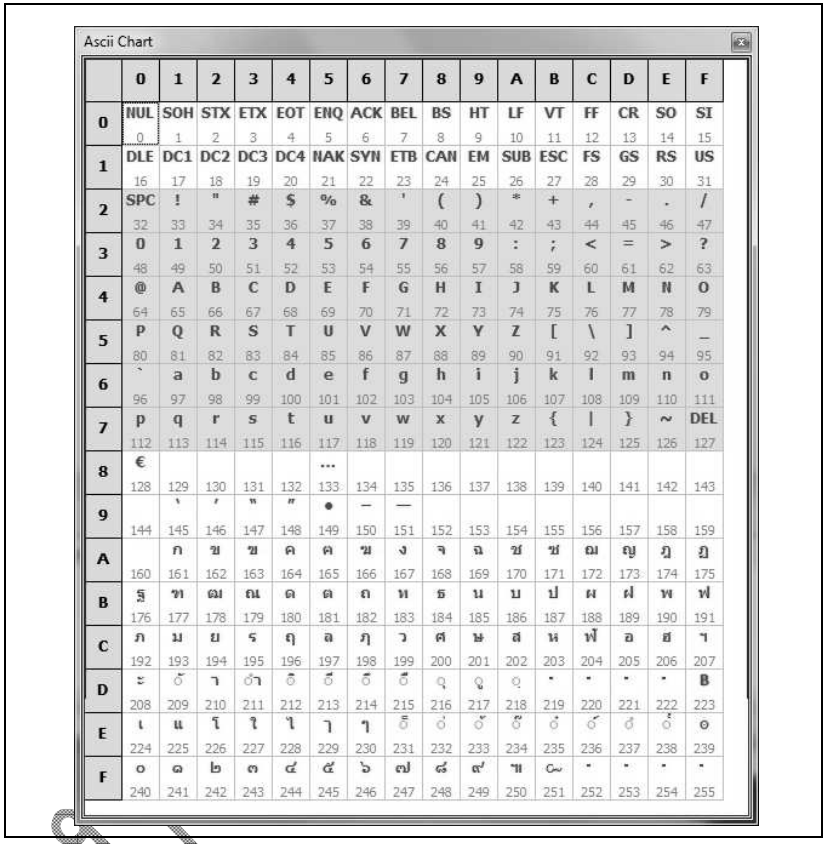
dim ชื่อตัวแปร **as string**[จำนวนอักขระ]

โปรแกรม mikroBasic PIC ได้เตรียมเครื่องมือสำหรับแสดงอักขระลำดับที่ 0 ถึง 255 หรือที่รู้จักกันในชื่อของรหัสแอสกี (ASCII) เอาไว้ให้ด้วยการเข้าที่เมนู Tools และเลือกรายการที่เขียนว่า Ascii chart (รูปที่ 4.1) หลังจากนั้นจะมีหน้าต่างของเครื่องมือ Ascii chart ปรากฏขึ้นมา (รูปที่ 4.2)

รูปที่ 4.1 เมนูเครื่องมือ Ascii chart ในเมนู Tools



รูปที่ 4.2 หน้าจอแสดงผลของเครื่องมือ ASCII Table



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL 0	SOH 1	STX 2	ETX 3	EOT 4	ENQ 5	ACK 6	BEL 7	BS 8	HT 9	LF 10	VT 11	FF 12	CR 13	SO 14	SI 15
1	DLE 16	DC1 17	DC2 18	DC3 19	DC4 20	NAK 21	SYN 22	ETB 23	CAN 24	EM 25	SUB 26	ESC 27	FS 28	GS 29	RS 30	US 31
2	SPC 32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
6	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	DEL 127
8	€ 128	129	130	131	132	… 133	134	135	136	137	138	139	140	141	142	143
9	144	‘ 145	’ 146	“ 147	” 148	• 149	– 150	— 151	152	153	154	155	156	157	158	159
A	160	ก 161	ข 162	ฃ 163	ค 164	ฅ 165	ฆ 166	ง 167	จ 168	ฉ 169	ช 170	ซ 171	ฌ 172	ญ 173	ฎ 174	ฏ 175
B	ฐ 176	ฑ 177	ฒ 178	ณ 179	ด 180	ต 181	ถ 182	ท 183	ธ 184	น 185	บ 186	ป 187	ผ 188	ฝ 189	พ 190	ฟ 191
C	ภ 192	ม 193	ย 194	ร 195	ฤ 196	ล 197	ฦ 198	ว 199	ศ 200	ษ 201	ส 202	ห 203	ฬ 204	อ 205	ฮ 206	ฯ 207
D	ะ 208	ั 209	า 210	ำ 211	ิ 212	ี 213	ึ 214	ื 215	ุ 216	ู 217	ฺ 218	- 219	- 220	- 221	- 222	฿ 223
E	เ 224	แ 225	โ 226	ใ 227	ไ 228	ๅ 229	ๆ 230	็ 231	่ 232	้ 233	๊ 234	๋ 235	์ 236	ํ 237	๎ 238	๏ 239
F	๐ 240	๑ 241	๒ 242	๓ 243	๔ 244	๕ 245	๖ 246	๗ 247	๘ 248	๙ 249	๚ 250	๛ 251	- 252	- 253	- 254	- 255

ตัวอย่างที่ 4.10 การใช้สายอักขระ

บรรทัด	โค้ด
1	program sample10
2	dim UsrName as String[20]
3	dim Prompt as String[10]
4	dim idx as byte
5	dim receive as byte
6	const unPrompt as String[10] = "Username ?"
7	main:
8	UART1_Init(9600)
9	Delay_ms(100)
10	Prompt = unPrompt
11	idx = 0
12	UsrName = ""
13	receive = 0
14	UART1_write_text(Prompt)
15	while ((idx < 20) and (receive <> 13))
16	if UART1_Data_Ready() = 1 then
17	receive = UART1_Read()
18	UsrName[idx] = receive
19	idx = idx + 1
20	end if
21	Wend
22	UART1_write_text(UsrName)
23	end.

จากตัวอย่าง 4.10 เป็นตัวอย่างการใช้งานตัวแปรสายอักขระ โดยตัวโปรแกรมจะแสดงข้อความว่า Username ? ไปที่พอร์ตอนุกรมลำดับที่ 1 (UART1) หลังจากนั้นรอรับค่าจากพอร์ตอนุกรมมาเก็บในตัวแปร receive และนำค่าที่เก็บในตัวแปรนี้ส่งต่อไปยังสายอักขระตัวที่ idx และทำจนกว่าจะครบ 20 อักขระ หรือมีการกดแป้น enter สุดท้ายจึงนำข้อความที่เก็บในสายอักขระที่ชื่อว่า UsrName ส่งออกไปที่พอร์ตอนุกรมอีกครั้งหนึ่ง ซึ่งขั้นตอนของการทำงานทั้งหมดเป็นดังนี้

บรรทัดที่ 2 เป็นการประกาศตัวแปรชื่อว่า UsrName ให้เป็นสายอักขระที่มีความยาว 20 ตัวอักษร

บรรทัดที่ 3 เป็นการประกาศตัวแปรชื่อว่า Prompt ให้เป็นสายอักขระที่มีความยาว 10 ตัวอักษร

บรรทัดที่ 4 เป็นการประกาศตัวแปรชื่อว่า idx ให้เป็นตัวแปรชนิด byte เพื่อใช้สำหรับนับจำนวนตัวอักษรที่รับเข้ามา และใช้เป็นตัวอ้างอิงตำแหน่งลำดับของตัวอักษรในตัวแปร UsrName

บรรทัดที่ 5 เป็นการประกาศตัวแปรชื่อว่า receive ให้เป็นตัวแปรชนิด byte เพื่อใช้สำหรับพักข้อมูลที่รับมาจากพอร์ตอนุกรม และนำค่าที่เก็บนี้ส่งไปเก็บในตัวแปร UsrName

บรรทัดที่ 6 เป็นการประกาศค่าคงที่ชื่อว่า unPrompt ให้เป็นค่าคงที่ชนิดสายอักขระที่ยาว 10 ตัวอักขระ และกำหนดค่าให้เป็นคำว่า Username ? เพื่อใช้สำหรับนำข้อความนี้ไปเก็บในตัวแปร Prompt และทำให้ได้เห็นว่ารูปแบบของการสร้างค่าคงที่ของสายอักขระนั้นเป็นเช่นไร

บรรทัดที่ 8 เป็นคำสั่งให้มีการเริ่มต้นการทำงานเพื่อทำการสื่อสารพอร์ตอนุกรมลำดับที่ 1 ด้วยอัตราการรับและส่ง หรือค่าบอร์ดเรต (baud rate) ที่ความเร็ว 9600 บิตต่อวินาที (bps)

บรรทัดที่ 9 เป็นคำสั่งให้มีการหน่วงเวลา 100 มิลลิวินาที

บรรทัดที่ 10 นำค่าที่เก็บในค่าคงที่ unPrompt ไปใส่ในตัวแปร Prompt

บรรทัดที่ 11 กำหนดให้ตัวแปร idx มีค่าเป็น 0

บรรทัดที่ 12 กำหนดให้ตัวแปร UsrName มีค่าเป็นค่าว่าง

บรรทัดที่ 13 กำหนดให้ตัวแปร receive มีค่าเป็น 0

บรรทัดที่ 14 เป็นคำสั่งส่งข้อความ Prompt ส่งไปที่พอร์ตอนุกรม

บรรทัดที่ 15 เป็นการวนรอบโดยมีเงื่อนไขว่าให้ทำถ้า idx น้อยกว่า 20 และ receive ไม่ใช่ค่า 13 (รหัสแอสกีของแป้น Enter)

บรรทัดที่ 16 เป็นการตรวจสอบว่ามีข้อมูลอยู่ที่พอร์ตอนุกรมหรือไม่ ถ้าใช่ให้ทำคำสั่งบรรทัดที่ 17 ถึงบรรทัดที่ 19

บรรทัดที่ 17 นำค่าจากพอร์ตอนุกรมมาเก็บในตัวแปร receive

บรรทัดที่ 18 นำค่าจากตัวแปร receive มาเก็บในตัวแปร UsrName ตัวที่ idx

บรรทัดที่ 19 เพิ่มค่าของตัวแปร idx ขึ้นมาอีก 1

บรรทัดที่ 22 เป็นคำสั่งนำค่า UsrName ส่งไปที่พอร์ตอนุกรม

15.ตัวชี้

ตัวชี้ (Pointer) คือ ชนิดของตัวแปรที่ทำหน้าที่เก็บค่าตำแหน่งของตัวแปรอื่น หรือเก็บค่าตำแหน่งของหน่วยความจำ ตัวแปรชนิดนี้มีประโยชน์ในการอ้างอิงค่าแบบทางอ้อม นั่นคือใช้ตัวแปรนี้เพื่ออ้างอิงค่าของตัวแปรอื่นหรือข้อมูล ณ ตำแหน่งใดๆ ในหน่วยความจำ การประกาศตัวแปรประเภทตัวชี้ต้องเพิ่มเครื่องหมาย ^ เอาไว้ด้านหน้าของชื่อประเภทข้อมูล เช่น ถ้าต้องการสร้างตัวแปรตัวชี้ที่ชี้ไปยังข้อมูลแบบ integer จะต้องใช้คำว่า ^integer และการอ้างอิงถึงชื่อตัวแปรจะต้องใส่เครื่องหมาย ^ ต่อท้ายชื่อ และจะใช้เครื่องหมาย @ ในการอ้างอิงถึงตำแหน่งของตัวแปรหรือโปรแกรมย่อย

รูปแบบของการประกาศตัวแปรชนิดตัวชี้เป็นดังนี้

dim ชื่อตัวแปร **as ^ชนิดของตัวแปรที่ต้องการอ้างอิงถึง**

รูปแบบของการหาค่าตำแหน่งของตัวแปรในหน่วยความจำ

@ชื่อตัวแปร หรือ ชื่อของฟังก์ชัน

รูปแบบของการกำหนดค่าให้แก่หน่วยความจำที่ตัวแปรชนิดตัวชี้อ้างอิงถึงเป็นดังต่อไปนี้

ชื่อตัวแปรชนิดตัวชี้^ = ค่าที่กำหนด

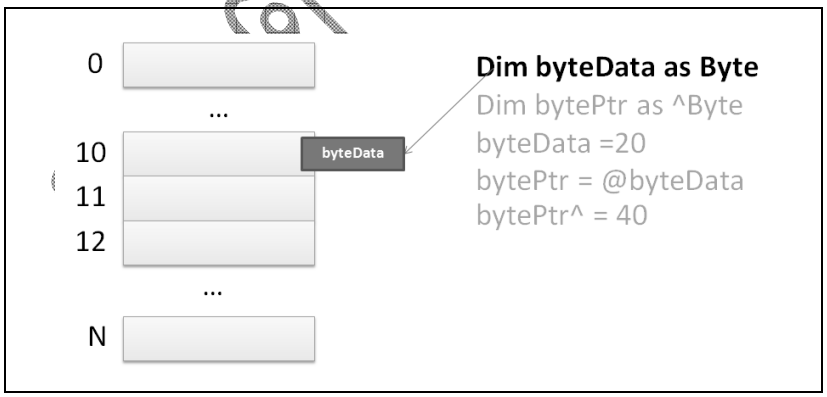
ตัวอย่างที่ 4.11 การใช้ตัวแปรชนิดตัวชี้

บรรทัด	โค้ด
1	program sample11
2	dim byteData as byte
3	dim bytePtr as ^byte
4	main:
5	byteData = 20
6	bytePtr = @byteData
7	bytePtr^ = 40
8	end.

จากตัวอย่างที่ 4.11 เป็นการใช้ตัวแปรตัวชี้เพื่อชี้ไปยังตัวแปร byteData หลังจากนั้นนำค่า 40 ไปใส่ในตัวโดยใช้วิธีการส่งข้อมูลทางอ้อมผ่านทางการอ้างอิงจากตัวแปรตัวชี้ ดังรายละเอียดของการทำงานต่อไปนี้

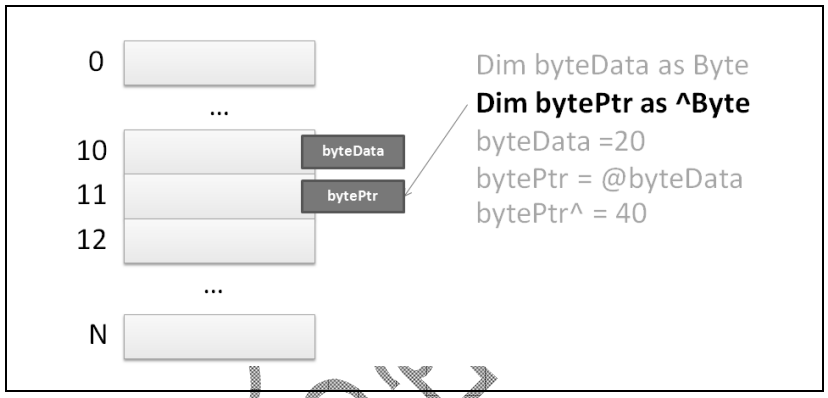
บรรทัดที่ 2 ประกาศตัวแปรชื่อว่า byteData ให้เป็นตัวแปรชนิด byte โดยในการทำงานจริงนั้นไมโครคอนโทรลเลอร์จะมองว่าตัวแปร byteData เป็นหน่วยความจำขนาด 1 ไบต์ และผู้เขียนสมมุติว่าตัวแปร byteData นี้อยู่ในหน่วยความจำ ณ ตำแหน่งที่ 10 (รูปที่ 4.3)

รูปที่ 4.3 ผลของการทำงานบรรทัดที่ 2 ของตัวอย่าง 4.11



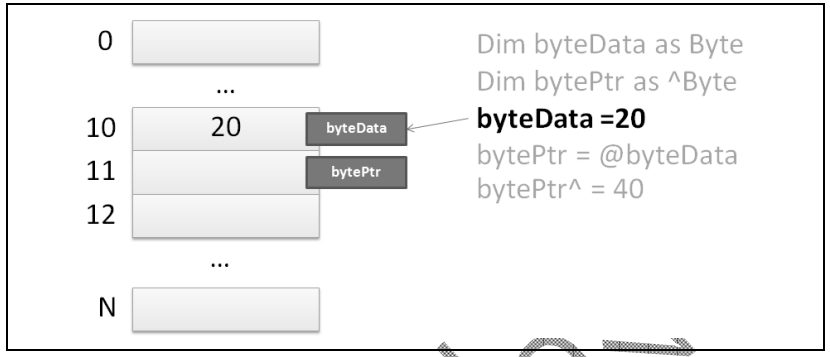
บรรทัดที่ 3 ประกาศตัวแปรชื่อว่า bytePtr ให้เป็นตัวแปรชนิดตัวชี้ที่ชี้ไปยังข้อมูลประเภท byte ซึ่งในการทำงานจะหมายความว่ามีการจองหน่วยความจำ (สมมุติว่าเป็นตำแหน่งที่ 11) เพื่อเก็บค่าของตัวแปร bytePtr ดังรูปที่ 4.4

รูปที่ 4.4 ผลของการทำงานบรรทัดที่ 3 ของตัวอย่าง 4.11



บรรทัดที่ 5 เป็นการสั่งให้ตัวแปรมีค่าเป็น 20 ซึ่งในการทำงานจริงของไมโครคอนโทรลเลอร์นั้นจะดูว่าตัวแปร byteData อยู่ ณ ตำแหน่งที่เท่าใดในหน่วยความจำ (สมมุติว่าเป็น 10) หลังจากนั้นไมโครคอนโทรลเลอร์จะนำค่า 20 ไปเก็บในหน่วยความจำที่ถูกมองว่าเป็นตัวแปร byteData ดังรูปที่ 4.5 ด้วยเหตุนี้จึงทำให้ผลของการทำงานที่ดังกล่าวนี้ จึงทำให้ตัวแปร byteData เก็บค่า 20

รูปที่ 4.5 ผลของการทำงานบรรทัดที่ 5 ของตัวอย่าง 4.11



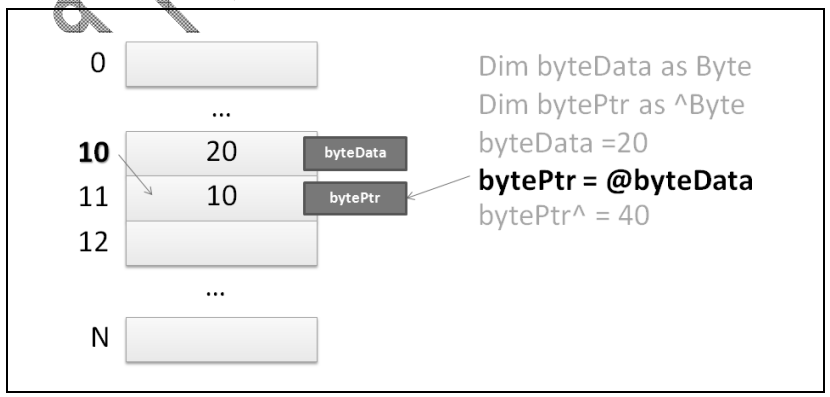
บรรทัดที่ 6 เป็นการสั่งให้นำค่าตำแหน่งของตัวแปร byteData ไปเก็บในตัวแปร bytePtr ซึ่งในการทำงานของไมโครคอนโทรลเลอร์นั้นจะมีทั้งสิ้น 3 ขั้นตอน คือ

ขั้นตอนที่ 1 หาค่าตำแหน่งของตัวแปร byteData ซึ่งมีค่าเป็น 10

ขั้นตอนที่ 2 หาค่าตำแหน่งของตัวแปร bytePtr ซึ่งมีค่าเป็น 11

ขั้นตอนที่ 3 นำค่าที่ได้จากขั้นตอนที่ 1 ไปเก็บในหน่วยความจำ ณ ตำแหน่งที่ได้จากขั้นตอนที่ 2 จึงทำให้ตัวแปร bytePtr เก็บค่า 10 ดังรูปที่ 4.6

รูปที่ 4.6 ผลของการทำงานบรรทัดที่ 6 ของตัวอย่าง 4.11



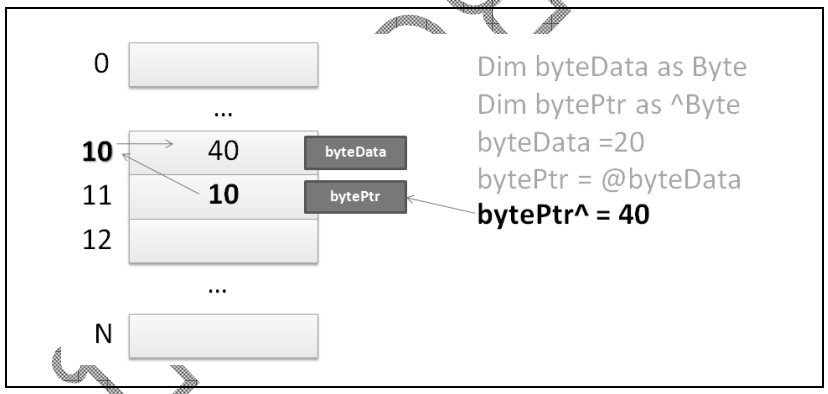
บรรทัดที่ 7 เป็นการนำค่า 40 ไปเก็บในหน่วยความจำที่ตัวแปร bytePtr อ้างอิงถึง ซึ่งในการทำงานจริงนั้นไมโครคอนโทรลเลอร์จะมองการทำงานเป็น 3 ขั้นตอน (รูปที่ 4.7) ดังนี้

ขั้นตอนที่ 1 หาค่าของตัวแปร bytePtr ซึ่งได้ค่าเป็น 11

ขั้นตอนที่ 2 หาค่าที่เก็บในตัวแปร bytePtr ซึ่งได้ค่าเป็น 10

ขั้นตอนที่ 3 นำค่า 40 ไปเก็บในหน่วยความจำ ณ ตำแหน่งที่ได้จากขั้นตอนที่ 2 ดังนั้น เมื่อดำเนินงานครบทั้ง 3 ขั้นตอน จึงทำให้ค่าของตัวแปร byteData ซึ่งเป็นตัวแปรที่อยู่ ณ ตำแหน่งที่ 10 ของหน่วยความจำมีค่าเป็น 40 จากเดิมที่มีค่าเป็น 20

รูปที่ 4.7 ผลของการทำงานบรรทัดที่ 7 ของตัวอย่าง 4.11



16. โครงสร้าง

โครงสร้าง (Structure) คือ ชนิดตัวแปรชนิดหนึ่งที่เกิดจากการนำตัวแปรหลากหลายชนิดมาอยู่รวมกัน และเรียกกลุ่มของตัวแปรเหล่านี้ว่าโครงสร้าง เช่น ในการเก็บข้อมูลของผู้ใช้ต้องเก็บชื่อและรหัสผ่าน เป็นต้น กรณีที่มีผู้ใช้เพียง 1 คน การเก็บไม่มีความยุ่งยาก เนื่องจากใช้ตัวแปรเพียง 2 ตัวแปรในการเก็บชื่อและรหัสผ่าน แต่ในกรณีที่เก็บข้อมูลผู้ใช้หลายคน เช่น เก็บข้อมูลของผู้ใช้ 10 คน ต้องสร้างตัวแปร 20 ตัวแปร แบ่งเป็นตัวแปรเก็บชื่อ 10 ตัวแปร และตัวแปรเก็บรหัสผ่าน 10 ตัวแปร ดังนี้

```
Dim UserName1, UserName2, UserName3, UserName4 as String[10]
Dim UserName5, UserName6, UserName7, UserName8 as String[10]
Dim UserName9, UserName10 as String[10] as String[10]
Dim UserPass1, UserPass2, UsePass3, UserPass4, UserPass5 as String[4]
Dim UserPass6, UserPass7, UserPass8, UserPass9, UserPass10 as String[4]
```

เมื่อเปลี่ยนวิธีการจัดเก็บจากตัวแปรปกติเป็นตัวแปรแบบแถวลำดับ จึงง่ายต่อการประกาศตัวแปร ดังนี้

```
Dim UserName as String[10][10]
Dim UserPass as String[10][4]
```

ในการใช้งานจะอาศัยการอ้างอิงว่า ค่าลำดับ 0 เป็นข้อมูลของผู้ใช้คนที่ 1 ค่าลำดับ 1 เป็นข้อมูลของผู้ใช้คนที่ 2 ตามลำดับ แต่ถ้ามีการจัดโครงสร้างของข้อมูลในรูปแบบของโครงสร้าง นั่นคือมองว่าผู้ใช้ประกอบไปด้วยข้อมูล 2 สิ่ง คือ ชื่อและรหัสผ่าน จะสามารถสร้างโครงสร้างของผู้ใช้ได้ดังนี้

```
structure UserInfo
   dim name as string[10]
   dim pass as string[4]
end structure
```


การนำโครงสร้าง UserInfo ไปใช้ประกาศเป็นชนิดของตัวแปร เพื่อสร้างตัวแปรชื่อ user เขียนโค้ดดังนี้

```
dim user as UserInfo
```

การกำหนดค่าให้กับชื่อผู้ใช้ และรหัสผ่านในตัวแปร user โดยให้ชื่อเป็น User1 และรหัสผ่านเป็น 1234 สามารถเขียนได้ดังนี้

```
user.name = "User1"
user.pass = "1234"
```

เมื่อต้องการใช้เก็บข้อมูลผู้ใช้จำนวน 5 คน ทำได้โดยการประกาศตัวแปรแถวลำดับประเภทโครงสร้าง UserInfo ดังนี้

```
dim userN as UserInfo[5]
```

ตัวอย่างการกำหนดข้อมูลชื่อและรหัสผ่านให้กับผู้ใช้คนที่ 3 ให้มีข้อมูลเป็น “Jaroot” และ “3456” ตามลำดับ เขียนโค้ดดังนี้

```
userN[2].name = "Jaroot"
userN[2].pass = "3456"
```

จากตัวอย่างที่ยกมาข้างต้น สรุปเป็นรูปแบบการกำหนดโครงสร้าง ดังนี้

```
structure ชื่อโครงสร้าง
    dim ชื่อตัวแปรที่1 as ชนิดของตัวแปร
    dim ชื่อตัวแปรที่2 as ชนิดของตัวแปร
    dim ชื่อตัวแปรที่3 as ชนิดของตัวแปร
    ...
    dim ชื่อตัวแปรที่N as ชนิดของตัวแปร
end structure
```

รูปแบบของการอ้างอิงถึงข้อมูลในโครงสร้างมีรูปแบบดังนี้

```
ชื่อตัวแปรชนิดโครงสร้าง.ชื่อตัวแปร
```

ตัวอย่างที่ 4.12 การใช้ตัวแปรชนิดโครงสร้าง

บรรทัด	โค้ด
1	program sample12
2	structure UserInfo
3	dim name as string[10]
4	dim pass as string[4]
5	end structure
6	dim user as UserInfo
7	dim userN as UserInfo[5]
8	dim username as string[10]
9	dim idxStr as string[2]
10	dim idx as byte
11	main:
12	user.name = "User1"
13	user.pass = "1234"
14	for idx=0 to 4
15	username = "UserN"
16	IntToStr(idx, idxStr)
17	strcat(username, idxStr)
18	userN[idx].name = username
19	userN[idx].pass = "1234"
20	next idx
21	end.

จากตัวอย่าง 4.12 เป็นการเขียนโปรแกรมเพื่อสร้างโครงสร้างชื่อว่า UserInfo และนำโครงสร้างมาประกาศเป็นตัวแปร 2 ตัวแปร คือ user และ userN โดยที่ตัวแปร userN ถูกประกาศเป็นตัวแปรชนิดแถวลำดับของโครงสร้าง UserInfo หลังจากนั้นเป็นการสั่งเพื่อกำหนดค่าให้กับตัวแปรทั้ง 2 ดังรายละเอียดต่อไปนี้

บรรทัดที่ 2-5 สร้างโครงสร้างชื่อว่า UserInfo โดยในโครงสร้างนี้มีข้อมูล 2 ชนิด คือ name และ pass โดยเป็นข้อมูลประเภทสายอักขระจำนวน 10 ตัวอักษร และสายอักขระจำนวน 4 ตัวอักษระตามลำดับ

บรรทัดที่ 6 เป็นการประกาศตัวแปรชื่อ user ให้เป็นตัวแปรโครงสร้างชนิด UserInfo

บรรทัดที่ 7 เป็นการประกาศตัวแปรชื่อ userN ให้เป็นตัวแปรแถวลำดับจำนวน 5 สมาชิก โดยที่สมาชิกแต่ละตัวเป็นโครงสร้างชนิด UserInfo

บรรทัดที่ 8 เป็นการประกาศตัวแปรชื่อ username เป็นตัวแปรชนิดสายอักขระยาว 10 ตัวอักษรสำหรับเป็นที่พักข้อมูลของชื่อผู้ใช้

บรรทัดที่ 9 เป็นการประกาศตัวแปรชื่อ idxStr เป็นตัวแปรชนิดสายอักขระยาว 2 ตัวอักษร สำหรับเก็บผลลัพธ์จากการแปลข้อมูลตัวเลข และนำค่านี้ไปใช้ต่อท้ายสายอักขระของคำว่า UserN ในตัวแปร username

บรรทัดที่ 10 เประกาศตัวแปรชื่อ idx เป็นตัวแปรชนิด byte สำหรับใช้ในการทำซ้ำและเป็นข้อมูลที่จะถูกแปลงให้เป็นสายอักขระเพื่อเก็บในตัวแปร idxStr

บรรทัดที่ 12 เป็นคำสั่งให้นำคำว่า User1 ไปเก็บในส่วนของ name ในตัวแปร user

บรรทัดที่ 13 เป็นคำสั่งให้นำคำว่า 1234 ไปเก็บในส่วนของ pass ในตัวแปร user

บรรทัดที่ 14-20 เป็นคำสั่งทำซ้ำด้วยการใช้ตัวแปร idx เป็นตัวแปรนับรอบ โดยตัวแปร idx จะมีค่าเริ่มต้นที่ 0 และสิ้นสุดที่ 4 จึงทำให้การวนรอบทำทั้งสิ้น 5 รอบ

บรรทัดที่ 15 เป็นการกำหนดให้ตัวแปร username มีค่าเป็นสายอักขระว่า UserN

บรรทัดที่ 16 เป็นการสั่งเพื่อแปลงค่าตัวเลข idx ให้เป็นสายอักขระ โดยนำผลการแปลงไปเก็บในตัวแปร idxStr

บรรทัดที่ 17 เป็นการสั่งเชื่อสายอักขระ โดยนำสายอักขระจาก idxStr มาต่อท้ายในสายอักขระ Username

บรรทัดที่ 18 เป็นการนำค่าจากตัวแปร username ไปเก็บในส่วนของ name ในตัวแปร userN ลำดับที่ idx

บรรทัดที่ 19 เป็นการนำค่าสายอักขระ 1234 ไปเก็บในส่วนของ pass ในตัวแปร userN ลำดับที่ idx

17. การสร้างชนิดตัวแปร

การสร้างชนิดตัวแปรเป็นการให้ผู้เขียนโปรแกรมสามารถสร้างชนิดตัวแปรขึ้นจากชนิดของตัวแปรมาตรฐานที่ได้กำหนดเอาไว้ ทั้งนี้เพื่อให้ผู้ใช้สามารถตั้งชื่อชนิดตัวแปรที่เข้าใจได้ง่ายขึ้น หรือใช้สำหรับลดจำนวนอักขระของชนิดตัวแปร ตัวอย่างเช่น ผู้เขียนโปรแกรมต้องการสร้างตัวแปรเพื่อเก็บค่าตำแหน่งจุดใดๆ ในพิกัด 2 มิติ ด้วยตัวแปรแถวลำดับ ซึ่งปกติจะประกาศตัวแปรได้ดังนี้

```
Dim P1 as word[2]
```

แต่ถ้าผู้เขียนโปรแกรมต้องการเจาะจงว่า P1 ใช้สำหรับเก็บข้อมูลจุดในพิกัด 2 มิติ กระทำได้ด้วยการสร้างชนิดตัวแปรขึ้นมาใหม่ เช่น

```
typedef Point2D as word[2]
```

หลังจากสร้างชนิดตัวแปรแล้วจึงประกาศตัวแปร P1 เพื่อให้เป็นชนิด Point2D ได้ดังนี้

```
Dim P1 as Point2D
```

ด้วยวิธีการประกาศชนิดตัวแปรจากชนิดตัวแปรที่ผู้เขียนสร้างขึ้นเองนั้น ทำให้อ่านโค้ดโปรแกรมได้ง่ายขึ้น เนื่องจากเมื่ออ่านจากการประกาศตัวแปรก็จะทราบในทันทีว่าตัวแปร P1 นั้นใช้สำหรับเก็บจุดในพิกัด 2 มิติ

จากตัวอย่างดังที่กล่าว รูปแบบของการประกาศเพื่อสร้างชนิดตัวแปร เขียนได้ในรูปแบบต่อไปนี้

typedef ชื่อชนิดของตัวแปรที่ต้องการสร้าง **as** ชนิดตัวแปร

ตัวอย่างที่ 4.13 การสร้างชนิดตัวแปรขึ้นมาเอง

บรรทัด	โค้ด
1	`program sampleTypeDef`
2	`typedef Point2D as word[2]`
3	`typedef Matrix2D as word[2][2]`
4	`dim P1 as Point2D`
5	`dim P2 as Point2D`
6	`dim MS as Matrix2D`
7	`main:`
8	`    P1[0] = 10`
9	`    P1[1] = 2`
10	`    MS[0][0] = 2`
11	`    MS[0][1] = 0`
12	`    MS[1][0] = 0`
13	`    MS[1][1] = 10`
14	`    P2[0] = P1[0]*MS[0][0] + P1[1]*MS[1][0]`
15	`    P2[1] = P1[0]*MS[0][1] + P1[1]*MS[1][1]`
16	`end.`

จากตัวอย่าง 4.13 เป็นการเขียนโปรแกรมเพื่อสร้างชนิดของตัวแปรชื่อว่า Point2D และ Matrix2D ขึ้นมาใช้งาน เพื่อคำนวณหาพิกัดของ P2 เมื่อ P1 ถูกสเกล (Scale) ค่าเป็น (2,10) ซึ่งการทำงานของโปรแกรมมีรายละเอียดดังนี้

บรรทัดที่ 2 เป็นการประกาศชนิดตัวแปรชื่อว่า Point2D ให้เป็นตัวแปรชนิดแถวลำดับจำนวน 2 สมาชิก โดยแต่ละสมาชิกเป็นข้อมูลแบบ word

บรรทัดที่ 3 เป็นการประกาศชนิดตัวแปรชื่อว่า Matrix ให้เป็นตัวแปรชนิดแถวลำดับจำนวน 4 สมาชิก คือ มี 2 แถว และแต่ละแถวมี 2 สดมภ์ โดยแต่ละสมาชิกเป็นข้อมูลแบบ word

บรรทัดที่ 4 เป็นการประกาศตัวแปร P1 ให้เป็นตัวแปรชนิด Point2D

บรรทัดที่ 5 เป็นการประกาศตัวแปร P2 ให้เป็นตัวแปรชนิด Point2D

บรรทัดที่ 6 เป็นการประกาศตัวแปร MS ให้เป็นตัวแปรชนิด Matrix2D

บรรทัดที่ 8 เป็นการกำหนดให้สมาชิกตัวแรกของ P1 ซึ่งใช้สำหรับเก็บค่าตำแหน่งในแกน X มีค่าเป็น 10

บรรทัดที่ 9 เป็นการกำหนดให้สมาชิกตัวที่ 2 ของ P1 ซึ่งใช้สำหรับเก็บค่าตำแหน่งในแกน Y มีค่าเป็น 2

บรรทัดที่ 10 เป็นการกำหนดให้สมาชิกสดมภ์แรกในแถวแรกของตัวแปร MS เก็บค่า 2

บรรทัดที่ 11 เป็นการกำหนดให้สมาชิกสดมภ์ที่ 2 ในแถวแรกของตัวแปร MS เก็บค่า 0

บรรทัดที่ 12 เป็นการกำหนดให้สมาชิกสดมภ์แรกในแถวที่ 2 ของตัวแปร MS เก็บค่า 0

บรรทัดที่ 13 เป็นการกำหนดให้สมาชิกสดมภ์ที่ 2 ในแถวที่ 2 ของตัวแปร MS เก็บค่า 10

บรรทัดที่ 14 เป็นการคำนวณหาค่าในแกน X ของตัวแปร P2 ซึ่งได้มาจากการรวมกันระหว่างสมาชิกตัวแรกของ P1 คูณกับสมาชิกสดมภ์ที่ 1 ในแถวที่ 1 ของตัวแปร MS บวกกับผลคูณระหว่างสมาชิกตัวที่ 2 ของ P1 คูณกับสมาชิกสดมภ์ที่ 1 ในแถวที่ 2 ของตัวแปร MS

บรรทัดที่ 15 เป็นการคำนวณหาค่าในแกน Y ของตัวแปร P2 ซึ่งได้มาจากการรวมกันระหว่างสมาชิกตัวแรกของ P1 คูณกับสมาชิกสดมภ์ที่ 2 ในแถวที่ 1 ของตัวแปร MS บวกกับผลคูณระหว่างสมาชิกตัวที่ 2 ของ P1 คูณกับสมาชิกสดมภ์ที่ 2 ในแถวที่ 2 ของตัวแปร MS

ดังนั้น เมื่อโปรแกรมในตัวอย่าง 4.13 ทำงานเสร็จจะทำให้ค่าของ P2 เป็น (20, 20)

18. ไดเร็คทีฟ

ไดเร็กทีฟ (Directive) หรือคำสั่งชี้แนะ คือ คำสั่งที่ใช้กำหนดการทำงานของตัวแปลภาษาให้ดำเนินงานตามที่ผู้เขียนโปรแกรมกำหนด ซึ่งใน mikroBasic PIC มีไดเร็กทีฟ 2 คำ คือ Absolute และ Org ซึ่งมีหน้าที่ดังตารางที่ 4.3

ตารางที่ 4.3 ไดเร็กทีฟของ mikroBasic PIC

ไดเร็กทีฟ	ความหมาย
Absolute	กำหนดตำแหน่งของตัวแปรในหน่วยความจำ
Org	กำหนดตำแหน่งของโปรแกรมย่อยในรอม

ตัวอย่างที่ 4.14 การใช้ไดเร็กทีฟ

```
บรรทัด  โค้ด
1       program sample14
2       dim myByte as byte absolute 0x28
3       dim myWord as word absolute 0x31
4
5       sub procedure myProc org $120
6       end sub
7
8       main:
9         goto main
10      end.
```

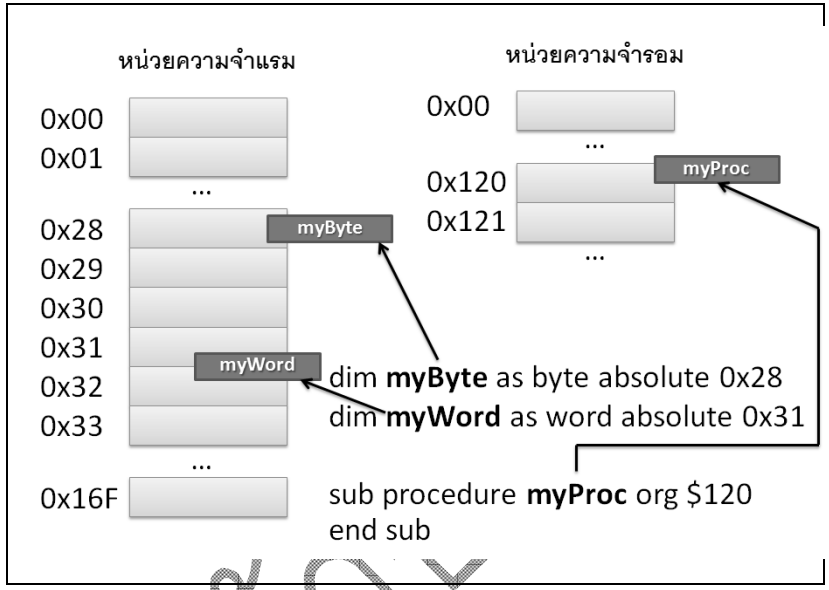
จากตัวอย่าง 4.14 เป็นตัวอย่างของการใช้ไดเร็กทีฟ absolute และ org เพื่อกำหนดให้ตัวแปรถูกสร้างในตำแหน่งที่ระบุ และกำหนดให้โปรแกรมย่อยที่ชื่อว่า myProc อยู่ในรอม ณ ตำแหน่งที่ 120_{16} เป็นต้นไป (ดูรูปที่ 4.8 ประกอบ) ซึ่งรายละเอียดของโปรแกรมเป็นดังนี้

บรรทัดที่ 2 เป็นการประกาศตัวแปรชื่อว่า myByte ให้เป็นตัวแปรชนิด byte และกำหนดให้อยู่ตำแหน่ง 28_{16} ของหน่วยความจำแรม

บรรทัดที่ 3 เป็นการประกาศตัวแปรชื่อว่า myWord ให้เป็นตัวแปรชนิด word และกำหนดให้อยู่ตำแหน่ง 31_{16} เป็นต้นไป จึงทำให้หน่วยความจำตำแหน่ง 31_{16} และ 32_{16} เป็นหน่วยความจำสำหรับเก็บค่าของตัวแปร myWord

บรรทัดที่ 5 เป็นการกำหนดให้โปรแกรมย่อยที่ชื่อว่า myProc ให้อยู่ในรอม ณ ตำแหน่ง 120_{16} เป็นต้นไป

รูปที่ 4.8 การใช้ไดเร็กทีฟ absolute กับ org



19. นิพจน์

นิพจน์ (Expression) คือ การแปลความข้อความที่ถูกเขียนในรูปแบบของสัญลักษณ์เพื่อหาผลการกระทำระหว่างกันของข้อความนั้น โดยนิพจน์นั้นเป็นได้ทั้งค่าคงที่ ตัวแปร ฟังก์ชัน หรือการกระทำระหว่างกันของนิพจน์ด้วยเครื่องหมายดำเนินการ เช่น การหาผลรวมของค่าที่เก็บในตัวแปร num1 และ num2 สามารถเขียนเป็นนิพจน์ num1 + num2 เป็นต้น

ในการเขียนนิพจน์นั้นจะประกอบไปด้วยค่าคงที่ ตัวแปร และเครื่องหมายดำเนินการ โดยจำนวนรูป หรือวิธีการเขียนนั้นขึ้นอยู่กับประเภทของเครื่องหมายดำเนินการนั้นๆ เช่น การเขียนนิพจน์ของค่าลบของตัวแปร num1 จะเขียนเป็น –num1 การเขียนส่วนกลับของตรรกะของตัวแปร num2 จะเขียนเป็น not num2 หรือการเลื่อนบิตของตัวแปร num3 ไปทางขวา 3 บิต จะเขียนเป็น num3 >> 3 เป็นต้น

ด้วยเหตุนี้สรุปได้ว่า รูปแบบของการเขียนนิพจน์มีด้วยกัน 5 รูปแบบคือ

1. การกำหนดค่า
2. ค่าคงที่/ตัวแปร/ฟังก์ชัน
3. นิพจน์ที่เครื่องหมายดำเนินการมีตัวถูกดำเนินการ 1 ตัว
4. นิพจน์ที่เครื่องหมายดำเนินการมีตัวถูกดำเนินการ 2 ตัว
5. นิพจน์ที่เกิดจากการนำทั้ง 3 รูปแบบเขียนผสมผสานกัน

รูปแบบของการกำหนดค่านั้น เป็นดังด้านล่าง ซึ่งหลักการกำหนดค่าในการเขียนโปรแกรมคอมพิวเตอร์จะแตกต่างจากเครื่องหมายเท่ากับ (=) ในทางคณิตศาสตร์ คือ ในทางคณิตศาสตร์นั้นมองว่าทางด้านซ้ายและด้านขวาของเครื่องหมายเท่ากับนั้นเป็นสิ่งที่มีค่าเท่ากัน แต่ในทางคอมพิวเตอร์นั้นมองว่า ให้หาผลลัพธ์ที่อยู่ทางขวา แล้วนำผลลัพธ์ที่ได้นั้นไปใส่ให้กับค่าทางซ้าย ด้วยเหตุนี้ ในการกำหนดค่านั้น ค่าทางซ้ายจึงต้องเป็นตัวแปร และทางด้านขวาเป็นนิพจน์

ตัวแปร = นิพจน์

รูปแบบของนิพจน์ในกรณีที่เครื่องหมายดำเนินการมีตัวถูกดำเนินการเพียง 1 ตัว เป็นดังนี้

เครื่องหมายดำเนินการ ตัวตัวถูกดำเนินการ

รูปแบบของนิพจน์ในกรณีที่เครื่องหมายดำเนินการมีตัวถูกดำเนินการ2 ตัว เป็นดังนี้

ตัวถูกดำเนินการทางซ้าย เครื่องหมายดำเนินการ ตัวถูกดำเนินการทางขวา

ตัวอย่าง 4.15

การเขียนนิพจน์ของค่าแรงดันมีค่าเท่ากับผลคูณของกระแสและค่าความต้านทาน สามารถเขียนได้ดังนี้

```
Volt = Ampere * Resistance
```

จากนิพจน์ด้านบนมีความหมายว่า ให้นำค่าที่เก็บในตัวแปร Ampere คูณ (คอมพิวเตอร์ใช้เครื่องหมาย * แทนเครื่องหมายคูณ) กับค่าที่เก็บในตัวแปร Resistance แล้วนำผลลัพธ์ที่ได้ไปเก็บไว้ในตัวแปร Volt

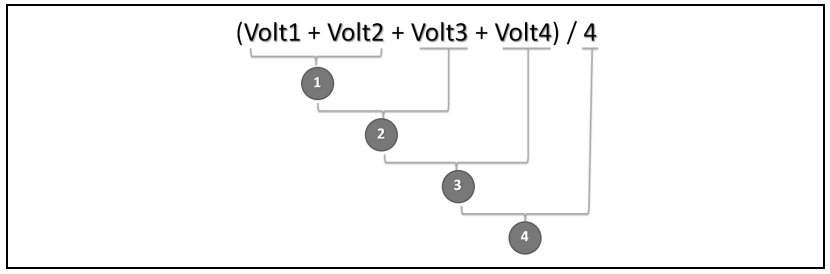
ตัวอย่าง 4.16

การเขียนนิพจน์ของการหาค่าเฉลี่ยของค่าแรงดัน 4 ค่า สามารถเขียนได้ดังนี้

```
AverageVolt = (Volt1+Volt2+Volt3+Volt4)/4
```

โดยที่นิพจน์ด้านบนนี้มีเครื่องหมายวงเล็บ เพื่อเป็นการบอกลำดับความสำคัญโดยให้กระทำในวงเล็บก่อน นั่นคือหาผลรวมระหว่างตัวแปร Volt1 กับ Volt2 หลังจากนั้นนำผลลัพธ์ที่ได้มารวมกับค่าในตัวแปร Volt3 สุดท้ายจึงนำผลลัพธ์จากการรวมกันมารวมกับค่าในตัวแปร Vol4 จึงได้ผลรวมทั้งหมดในเครื่องหมายวงเล็บ แล้วค่อยนำผลลัพธ์จากการหาผลรวมมาหาร (คอมพิวเตอร์ใช้เครื่องหมาย / แทนการหาร) ด้วย 4 จากตัวอย่างนี้พบว่า การหาค่านั้นจะกระทำทีละชุดข้อมูลโดยดูจากเครื่องหมายดำเนินการ ดังนั้น นิพจน์ด้านบนนั้นมีขั้นตอนการทำงานทั้งหมด 4 ครั้ง ดังรูปที่ 4.9

รูปที่ 4.9 ลำดับการประมวลผลนิพจน์ (Volt1+Volt2+Volt3+Volt4)/4

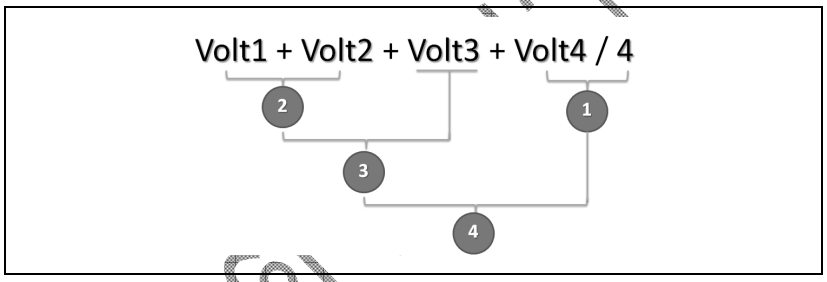


แต่ถ้าเขียนนิพจน์ใหม่เป็นดังต่อไปนี้

```
AverageVolt = Volt1+Volt2+Volt3+Volt4/4
```

จะทำให้ผลของการทำงานผิดพลาด เนื่องจากคอมพิวเตอร์มองว่าจะต้องนำค่าจากตัวแปร Volt4 มาหาร ด้วย 4 แล้วค่อยนำผลลัพธ์จากการหารนี้พักเอาไว้ชั่วคราว หลังจากนั้นไปหาผลรวมของค่าในตัวแปร Volt1 กับ Volt2 แล้วนำผลรวมที่ได้นี้ไปรวมกับค่าในตัวแปร Volt3 สุดท้ายจึงนำผลลัพธ์จากการรวมเข้ากับผลลัพธ์ที่เกิดจากการหาร Volt4 ด้วย 4 และสุดท้ายนำผลลัพธ์สุดท้ายไปเก็บในตัวแปร AverageVolt (ดูรูปที่ 4.10 ประกอบ)

รูปที่ 4.10 ลำดับการประมวลผลนิพจน์ Volt1+Volt2+Volt3+Volt4/4



20. เครื่องหมายดำเนินการ

จากหัวข้อก่อนหน้านี้ที่กล่าวถึงนิพจน์จะเห็นว่า การใช้นิพจน์ประกอบด้วยเครื่องหมายดำเนินการชนิดต่างๆ ดังนั้น ผู้เขียนโปรแกรมจึงจำเป็นที่จะต้องรู้จักกับเครื่องหมายดำเนินการต่างๆ ที่ภาษาเขียนโปรแกรมนั้นๆ สนับสนุน จึงเป็นพื้นฐานในการเขียนนิพจน์ให้ได้ผลลัพธ์ออกมาตามที่ต้องการหรือออกมาตรงตามกับเป้าหมายที่กำหนดไว้

เครื่องหมายดำเนินการ (Operator) คือ เครื่องหมายที่ใช้สื่อแทนการกระทำสิ่งใดสิ่งหนึ่ง ซึ่งในภาษาเบสิกนั้นมีเครื่องหมายดำเนินการด้วยกัน 4 ด้าน คือ เครื่องหมายดำเนินการด้านคณิตศาสตร์ เครื่องหมายดำเนินการด้านการเปรียบเทียบ เครื่องหมายดำเนินการเกี่ยวกับบิต และเครื่องดำเนินการด้านตรรกะ

20.1 ลำดับความสำคัญ

เมื่อผู้เขียนโปรแกรมนำเครื่องหมายดำเนินการทั้ง 4 ด้านมาเขียนเป็นนิพจน์ สิ่งหนึ่งที่เป็นปัญหาใหญ่คือ จะทราบได้อย่างไรว่า ตัวแปลภาษาจะจับคู่การประมวลผลกับนิพจน์ใดก่อน ด้วยเหตุนี้ ภาษาเบสิกของ mikroBasic PIC จึงได้วางข้อกำหนดลำดับความสำคัญของเครื่องหมายดำเนินการเอาไว้ดังตาราง 4.4 คือ

ตารางที่ 4.4 ลำดับความสำคัญของเครื่องหมายดำเนินการ

ลำดับความสำคัญ	จำนวนตัวถูกกระทำ	เครื่องหมาย	ความเชื่อมโยง
4	1	@ not + -	ขวาไปซ้าย
3	2	* / div mod and << >>	ซ้ายไปขวา
2	2	+ - or xor	ซ้ายไปขวา
1	2	= <> < > <= >=	ซ้ายไปขวา

จากข้อกำหนดของตาราง 4.4 จะได้ว่า ถ้าเครื่องหมายดำเนินการใดมีค่าของลำดับความสำคัญสูงกว่าจะถูกเลือกมาประมวลผลก่อน และถ้าเป็นเครื่องหมายที่มีลำดับความสำคัญเท่าเทียมกัน จะพิจารณาตามความเชื่อมโยง นั่นคือ ถ้ากำหนดความเชื่อมโยงเป็นขวาไปซ้ายจะหมายความว่าทำเครื่องหมายดำเนินการทางด้านขวาก่อนเครื่องหมายดำเนินการด้านซ้าย และในทางกลับกัน ถ้าความเชื่อมโยงกำหนดเป็นซ้ายไปขวาจะเป็นการเลือกทำเครื่องหมายดำเนินการที่อยู่ทางด้านซ้ายก่อนเครื่องหมายดำเนินการด้านขวา

20.2 เครื่องหมายดำเนินการด้านคณิตศาสตร์

เครื่องหมายดำเนินการด้านคณิตศาสตร์ (Arithmetic Operator) คือ เครื่องหมายที่ใช้แทนการกระทำทางคณิตศาสตร์อันได้แก่ การบวก การลบ การคูณ การหาร การหารโดยปัดค่า และการหาเศษจากการหาร โดยใช้เครื่องหมายแทนการดำเนินการดังตารางที่ 4.5

ตารางที่ 4.5 เครื่องหมายดำเนินการด้านคณิตศาสตร์

เครื่องหมาย	ความหมาย	ลำดับความสำคัญ
+	การบวก	2
-	การลบ	2
*	การคูณ	3
/	การหาร	3
Div	การหารปัดเศษ	3
Mod	การหาเศษจากการหาร	3

รูปแบบของเครื่องหมายดำเนินการประเภทนี้ คือ

นิพจน์ทางซ้าย เครื่องหมาย นิพจน์ทางขวา

นั่นหมายความว่า ไมโครคอนโทรลเลอร์จะนำนิพจน์ทางซ้ายเป็นตัวถูกกระทำ และนำนิพจน์ทางขวา เป็นตัวกระทำ โดยดำเนินการตามเครื่องหมายดำเนินการที่กำหนด

ตัวอย่าง 4.14 เป็นตัวอย่างการใช้เครื่องหมายดำเนินการทางคณิตศาสตร์เพื่อคำนวณค่าพลังงาน (Power) เฉลี่ยจากค่าของแรงดันและค่าความต้านทานจำนวน 3 ชุด มีสมการของการหาค่าพลังงานเป็นดังนี้

$$Power = \frac{V^2}{R}$$

จากสมการหาค่าพลังงานสามารถแปลงเป็นนิพจน์ในภาษาเบสิกดังนี้

```
val_p = (val_v * val_v) / val_r
```

ดังนั้น ถ้าเป็นการหาค่าเฉลี่ยจากข้อมูล 3 ชุด จึงสามารถดังนี้

```
val_p[0] = (val_v[0] * val_v[0]) / val_r[0]
val_p[1] = (val_v[1] * val_v[1]) / val_r[1]
val_p[2] = (val_v[2] * val_v[2]) / val_r[2]
val_p_avg = (val_p[0]+val_p[1]+val_p[2]) / 3
```

ตัวอย่างที่ 4.17 การคำนวณหาค่าพลังงานจากแรงดันและความต้านทาน

บรรทัด	โค้ด
1	program sample15
2	dim val_v as float[3]
3	dim val_r as float[3]
4	dim val_p as float[3]
5	dim val_p_avg as float
6	main:
7	val_v[0] = 2.4
8	val_v[1] = 4.7
9	val_v[2] = 4.9
10	val_r[0] = 100
11	val_r[1] = 330
12	val_r[2] = 500
13	val_p[0] = (val_v[0]*val_v[0])/val_r[0]
14	val_p[1] = (val_v[1]*val_v[1])/val_r[1]
15	val_p[2] = (val_v[2]*val_v[2])/val_r[2]
16	val_p_avg = (val_p[0]+val_p[1]+val_p[2])/3
17	end.

รายละเอียดการทำงานของโปรแกรมตัวอย่าง 4.17 เป็นดังนี้

บรรทัดที่ 2 ประกาศตัวแปรชื่อ val_v สำหรับเก็บค่าแรงดัน โดยกำหนดเป็นแถวลำดับของตัวแปรชนิด float จำนวน 3 สมาชิก

บรรทัดที่ 3 ประกาศตัวแปรชื่อ val_r สำหรับเก็บค่าความต้านทานโดยกำหนดเป็นแถวลำดับของตัวแปรชนิด float จำนวน 3 สมาชิก

บรรทัดที่ 4 ประกาศตัวแปรชื่อ val_p สำหรับเก็บค่าพลังงาน โดยกำหนดเป็นแถวลำดับของตัวแปรชนิด float จำนวน 3 สมาชิก

บรรทัดที่ 5 ประกาศตัวแปรชื่อ val_p_avg สำหรับเก็บค่าพลังงานเฉลี่ย โดยกำหนดเป็นตัวแปรชนิด float

บรรทัดที่ 7 กำหนดให้สมาชิกตัวที่ 1 ของตัวแปร val_v มีค่าเป็น 2.4

บรรทัดที่ 8 กำหนดให้สมาชิกตัวที่ 2 ของตัวแปร val_v มีค่าเป็น 4.7

บรรทัดที่ 9 กำหนดให้สมาชิกตัวที่ 3 ของตัวแปร val_v มีค่าเป็น 4.9

บรรทัดที่ 10 กำหนดให้สมาชิกตัวที่ 1 ของตัวแปร val_r มีค่าเป็น 100

บรรทัดที่ 11 กำหนดให้สมาชิกตัวที่ 2 ของตัวแปร val_r มีค่าเป็น 330

บรรทัดที่ 12 กำหนดให้สมาชิกตัวที่ 3 ของตัวแปร val_r มีค่าเป็น 500

บรรทัดที่ 13 กำหนดให้สมาชิกตัวที่ 1 ของตัวแปร val_p เก็บผลจากการคำนวณค่าพลังงานจากข้อมูลค่าแรงดันและค่าความต้านทานชุดที่ 1

บรรทัดที่ 14 กำหนดให้สมาชิกตัวที่ 2 ของตัวแปร val_p เก็บผลจากการคำนวณค่าพลังงานจากข้อมูลค่าแรงดันและค่าความต้านทานชุดที่ 2

บรรทัดที่ 15 กำหนดให้สมาชิกตัวที่ 3 ของตัวแปร val_p เก็บผลจากการคำนวณค่าพลังงานจากข้อมูลค่าแรงดันและค่าความต้านทานชุดที่ 3

บรรทัดที่ 16 กำหนดให้ตัวแปร val_p_avg เก็บค่าเฉลี่ยของค่าพลังงาน ซึ่งหาได้จากการนำค่าผลรวมของพลังงานทั้ง 3 ชุด หารด้วย 3

ตัวอย่างที่ 4.18 คำนวณจำนวนรอบการหมุนและตำแหน่งที่เข็มนาฬิกาชี้

บรรทัด	โค้ด
1	program sample16
2	dim clock_goal as byte
3	dim clock_cnt as byte
4	dim clock_pos as byte
5	main:
6	clock_goal = 67
7	clock_cnt = clock_goal div 12
8	clock_pos = clock_goal mod 12
9	end.

ตัวอย่าง 4.18 เป็นการคำนวณค่าจำนวนรอบของการหมุนของนาฬิกาและตำแหน่งของเข็มชั่วโมง เมื่อนาฬิกาเดินไปได้ 67 ชั่วโมง

วิธีการคำนวณหาจำนวนรอบการเดินของนาฬิกา คือ การนำจำนวนชั่วโมงหารแบบปัดเศษด้วย 12 ซึ่งสาเหตุที่หารด้วย 12 เนื่องจากการเดินครบ 1 รอบนั้นใช้เวลา 12 ชั่วโมง ด้วยเหตุนี้เมื่อต้องเดิน 67 ชั่วโมง จำนวนรอบการของเดินจึงมีค่าเท่ากับ 67 div 12 หรือ 5 รอบ

วิธีการคำนวณหาตำแหน่งของเข็มนาฬิกาเมื่อเดินไปได้ 67 ชั่วโมงนั้นคำนวณได้ด้วยการนำค่าชั่วโมงหารเพื่อหาเศษด้วย 12 จึงได้ผลลัพธ์เท่ากับ 67 mod 12 หรือ 7

ดังนั้น เมื่อนาฬิกาเดินไป 67 ชั่วโมง จะทำให้เข็มหมุน 5 รอบ และหยุดอยู่ที่เลข 7

รายละเอียดการทำงานของโปรแกรมตัวอย่าง 4.18 เป็นดังนี้

บรรทัดที่ 2 ประกาศตัวแปรชื่อ clock_goal สำหรับเก็บค่าชั่วโมงการเดินของนาฬิกา โดยกำหนด เป็นตัวแปรชนิด byte

บรรทัดที่ 3 ประกาศตัวแปรชื่อ clock_cnt สำหรับเก็บจำนวนรอบการหมุนของนาฬิกา โดยกำหนดเป็นตัวแปรชนิด byte

บรรทัดที่ 4 ประกาศตัวแปรชื่อ clock_pos สำหรับเก็บค่าตำแหน่งที่เข็มนาฬิกาหยุด โดยกำหนดเป็นตัวแปรชนิด byte

บรรทัดที่ 6 กำหนดให้ตัวแปร clock_goal มีค่าเป็น 67

บรรทัดที่ 7 คำนวณค่าจำนวนรอบของการหมุนของนาฬิกา ด้วยการนำค่า clock_goal หารแบบปัดเศษด้วย 12

บรรทัดที่ 8 คำนวณตำแหน่งที่เข็มนาฬิกาหยุด ด้วยการนำค่า clock_goal หารเพื่อหาเศษด้วย 12

20.3 เครื่องหมายดำเนินการด้านการเปรียบเทียบ

เครื่องหมายดำเนินการด้านการเปรียบเทียบ (Relational Operators) คือ เครื่องหมายดำเนินการที่แทนการเปรียบเทียบระหว่างนิพจน์ที่เป็นตัวกระทำกับนิพจน์ที่เป็นตัวถูกกระทำด้วยเครื่องหมายที่กำหนดดังตารางที่ 4.6 ซึ่งผลลัพธ์ที่ได้จากการเปรียบเทียบนั้นจะมีคำตอบเพียง 2 ประเภท อย่างใดอย่างหนึ่งระหว่าง **จริง** (true) กับ **เท็จ** (false) เท่านั้น

ตารางที่ 4.6 เครื่องหมายดำเนินการด้านการเปรียบเทียบ

เครื่องหมาย	ความหมาย	ลำดับความสำคัญ
=	เท่ากัน	1
<>	ไม่เท่ากัน	1
>	มากกว่า	1
<	น้อยกว่า	1
>=	มากกว่าหรือเท่ากัน	1
<=	น้อยกว่าหรือเท่ากัน	1

ตัวอย่างที่ 4.19 การใช้เครื่องหมายดำเนินการด้านการเปรียบเทียบ

บรรทัด	โค้ด
1	`program sample17`
2	`dim val1, val2, val3, val4 as byte`
3	`main:`
4	`    val1 = 20`
5	`    val2 = val1 >= 10`
6	`    val3 = 32`
7	`    val4 = ((val1 < val3) <> val2)`
8	`    val2 = (val4 * val1 >= val3)`
9	`    val3 = (val2 = val4)`
10	`end.`

จากตัวอย่าง 4.19 เป็นตัวอย่างการใช้เครื่องหมายดำเนินการด้านการเปรียบเทียบ โดยตัวโปรแกรมมีรายละเอียดของการทำงานดังนี้

บรรทัดที่ 2 เป็นการประกาศตัวแปรชื่อว่า val1 val2 val3 และ val4 โดยทุกตัวเป็นตัวแปร ประเภท Byte

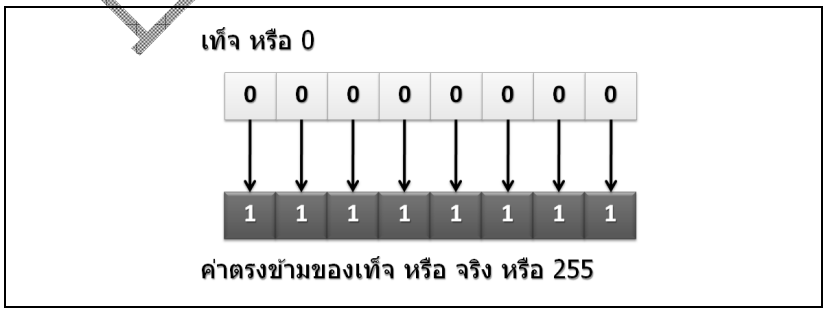
บรรทัดที่ 4 กำหนดให้ val1 มีค่าเป็น 20

บรรทัดที่ 5 กำหนดให้ val2 เก็บผลลัพธ์ของการเปรียบเทียบว่า val1 นั้นมากกว่าหรือเท่ากับ 10 หรือไม่ ซึ่งได้ผลลัพธ์เป็น จริง หรือ 255

หมายเหตุ

สาเหตุที่ทำให้ตัวแปร val2 มีค่าเป็น 255 เนื่องจาก mikroBasic มองค่าเท็จเป็น 0 ดังนั้นค่าตรงข้ามของ 0 จึงเป็น 255 ดังรูปที่ 4.11

รูปที่ 4.11 ค่าตรงข้ามของเท็จ



บรรทัดที่ 6 กำหนดให้ val3 มีค่าเป็น 32

บรรทัดที่ 7 กำหนดให้ val4 เก็บผลลัพธ์ของการเปรียบ โดยในขั้นตอนแรกนั้นจะเปรียบเทียบว่า val1 น้อยกว่า val3 นั้นเป็นจริงหรือไม่ หากได้ผลเป็นจริงจะเปรียบเทียบต่อว่าไม่เท่ากันกับค่าใน val2 ซึ่งเก็บค่าจริง ใช่หรือไม่ จึงได้คำตอบเป็นเท็จ ดังนั้น val4 จึงมีค่าเป็น 0 หรือเป็นเท็จ

บรรทัดที่ 8 กำหนดให้ val2 เก็บผลลัพธ์ของการดำเนินการ 2 อย่าง คือ นำค่าใน val4 คูณกับค่าใน val1 ซึ่งได้ผลของการคูณเป็น 0 และนำผลจากการคูณไปเปรียบเทียบว่ามีค่ามากกว่าหรือเท่ากับค่าที่เก็บใน val3 หรือไม่ จึงได้ผลลัพธ์ว่าเป็นเท็จ ดังนั้น val2 จึงมีค่าเป็นเท็จ หรือ 0

บรรทัดที่ 9 กำหนดให้ val3 เก็บผลของการเปรียบเทียบค่าระหว่าง val2 กับ val4 นั้นมีค่าเท่ากันหรือไม่ ซึ่งจะได้ว่าตัวแปร val2 มีค่าเป็นเท็จ และตัวแปร val4 มีค่าเป็นเท็จ ดังนั้น ค่าในตัวแปรทั้งสองนั้นเท่ากันจึงทำให้ตัวแปร val3 มีค่าเป็น จริง

เมื่อโปรแกรมตัวอย่างที่ 4.19 ทำงานเสร็จสิ้นจะได้ผลลัพธ์ของค่าที่เก็บในตัวแปรแต่ละตัวดังนี้

val1 มีค่าเป็น 20

val2 มีค่าเป็น เท็จ

val3 มีค่าเป็น จริง

val4 มีค่าเป็น เท็จ

20.4 เครื่องหมายดำเนินการเกี่ยวกับบิต

เครื่องหมายดำเนินการกับบิต (Bitwise Operators) คือ เครื่องหมายดำเนินการที่ใช้สำหรับการกระทำระหว่างนิพจน์ตที่ถูกกระทำด้วยนิพจน์ตัวกระทำด้วยเครื่องหมายดังตารางที่ 4.7

ตารางที่ 4.7 เครื่องหมายดำเนินการเกี่ยวกับบิต

เครื่องหมาย	ความหมาย	ลำดับความสำคัญ
And	กระทำบิตแบบการ and	3
Or	กระทำบิตแบบการ or	2
Xor	กระทำบิตแบบการ xor	2
Not	กระทำบิตแบบ 1 คอมพลีเมนต์	4
<<	กระทำเลื่อนบิตไปทางซ้าย	3
>>	กระทำเลื่อนบิตไปทางขวา	3

การกระทำบิตแบบการแอนด์ (And) มีรูปแบบของผลลัพธ์ดังตารางที่ 4.8 จะสังเกตเห็นว่าการที่จะได้ผลลัพธ์ของบิตเป็นค่า 1 นั้น บิตของตัวถูกกระทำและของตัวกระทำต้องเป็น 1 ทั้งคู่

ตารางที่ 4.8 ค่าความจริงของการกระทำแบบแอนด์

ตัวถูกกระทำ	ตัวกระทำ	ตัวถูกกระทำ **and** ตัวกระทำ
1	1	1
1	0	0
0	1	0
0	0	0

การกระทำบิตแบบการออร์ (Or) มีรูปแบบของผลลัพธ์ดังตารางที่ 4.9 จะเห็นว่าการที่ได้ผลลัพธ์ของบิตเป็นค่า 0 นั้นมีเพียง 1 กรณี คือ บิตของตัวถูกกระทำและของตัวกระทำต้องเป็น 0 ทั้งคู่

ตารางที่ 4.9 ค่าความจริงของการกระทำแบบออร์

ตัวถูกกระทำ	ตัวกระทำ	ตัวถูกกระทำ **or** ตัวกระทำ
1	1	1
1	0	1
0	1	0
0	0	0

การกระทำบิตแบบการเอ็กซ์คลูซีฟออร์ (Xor) มีรูปแบบของผลลัพธ์ดังตารางที่ 4.10 จะเห็นว่า การที่ได้ผลลัพธ์ของบิตเป็นค่า 1 เมื่อค่าบิตของตัวถูกกระทำ กับตัวกระทำนั้นมีความแตกต่างกันและได้ผลลัพธ์เป็น 0 เมื่อค่าบิตของตัวถูกกระทำและตัวกระทำเหมือนกัน

ตารางที่ 4.9 ค่าความจริงของการกระทำแบบเอ็กซ์คลูซีฟออร์

ตัวถูกกระทำ	ตัวกระทำ	ตัวถูกกระทำ **xor** ตัวกระทำ
1	1	0
1	0	1
0	1	1
0	0	0

การกระทำบิตแบบการนอต (Not) มีรูปแบบของผลลัพธ์ดังตารางที่ 4.10 ซึ่งการทำ 1 คอมพลีเมนต์ (1-Complement) คือ การกลับค่าของบิตของตัวถูกกระทำ และรูปแบบของการใช้นอต เป็นแบบมีตัวถูกกระทำ 1 ตัว คือ

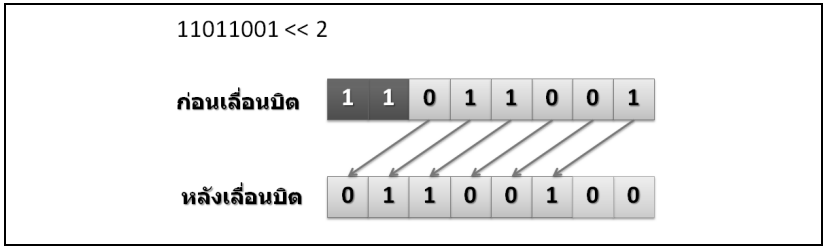
Not ตัวถูกกระทำ

ตารางที่ 4.10 ค่าความจริงของการกระทำแบบนอต

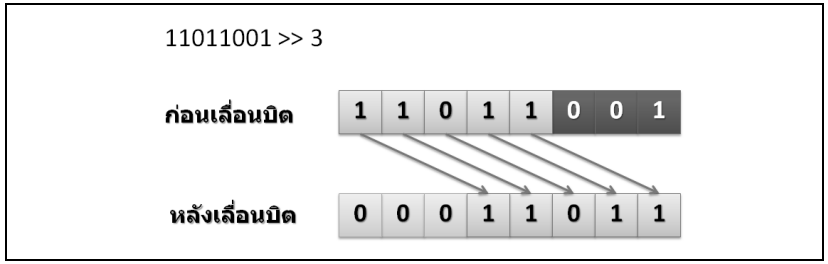
ตัวกระทำ	**Not** ตัวกระทำ
1	0
0	1

การเลื่อนบิต (Shift Bit) มีด้วยกัน 2 วิธี คือ การเลื่อนบิตไปทางซ้ายและเลื่อนบิตไปทางขวา โดยหลักของการเลื่อนบิต คือ นำบิตทางด้านหนึ่งออก และเติมบิตที่เป็นค่า 0 เข้าไปแทนในอีกด้านหนึ่ง ตัวอย่างเช่น การเลื่อนบิตของค่าตัวเลข 11011001_2 ไปทางซ้ายจำนวน 2 บิต จะทำให้บิต 11 ของทางซ้ายหายไป และมี 00 มาอยู่ทางด้านขวา ดังรูป 4.13 และการเลื่อนบิตของตัวเลขชุดเดียวกันไปทางขวา 3 บิต จะหมายความว่าบิตที่มีค่า 001 ที่อยู่ทางด้านขวาจะหายไป และจะมีบิต 000 มาอยู่ทางด้านซ้าย ดังรูป 4.14 เป็นต้น

รูปที่ 4.12 ผลของการเลื่อนบิตของเลข 11011001_2 ไปทางซ้าย 2 บิต



รูปที่ 4.13 ผลของการเลื่อนบิตของเลข 11011001_2 ไปทางขวา 3 บิต



ตัวอย่างที่ 4.20 การใช้เครื่องหมายดำเนินการเกี่ยวกับบิต

บรรทัด	โค้ด
1	`program sample18`
2	`dim val1, val2, val3, val4 as byte`
3	`main:`
4	`  val1 = 0xAB`
5	`  val2 = 0xC9`
6	`  val3 = val1 xor val2`
7	`  val4 = val3 xor val2`
8	`  val3 = val1 and val2`
9	`  val4 = val1 or val2`
10	`  val3 = val1 and not val4`
11	`  val1 = 0x01`
12	`  val2 = val1 << 1`
13	`  val3 = val1 << 2`
14	`  val4 = val3 >> 1`
15	`  val1 = val3 >> 2`
16	`end.`

ตัวอย่าง 4.20 เป็นตัวอย่างโปรแกรมที่เกี่ยวกับการนำเครื่องหมายดำเนินการเกี่ยวกับบิตมาใช้งาน ซึ่งรายละเอียดของการทำงานเป็นดังนี้

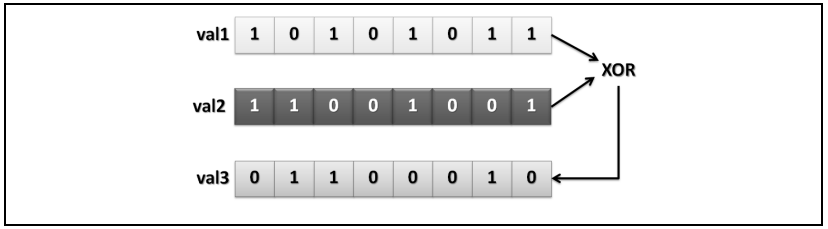
บรรทัดที่ 2 เป็นการประกาศตัวแปรชื่อ val1 val2 val3 และ val4 โดยแต่ละตัวแปรเป็นตัวแปรชนิด byte

บรรทัดที่ 4 กำหนดให้ตัวแปร val1 มีค่าเป็น AB_{16} หรือ 10101011_2

บรรทัดที่ 5 กำหนดให้ตัวแปร val2 มีค่าเป็น $C9_{16}$ หรือ 11001001_2

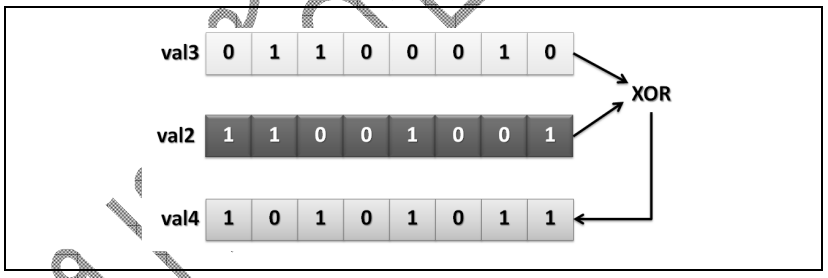
บรรทัดที่ 6 กำหนดให้ตัวแปร val3 เก็บผลลัพธ์ของการกระทำเอ็กซ์คลูซีฟออร์ระหว่างค่าในตัวแปร val1 ซึ่งมีค่าเป็น AB_{16} กับค่าในตัวแปร val2 ซึ่งมีค่าเป็น $C9_{16}$ จึงทำให้ผลลัพธ์ออกเป็น 62_{16} หรือ 01100010_2 ดังรูปที่ 4.14

รูปที่ 4.14 ผลของ 10101011_2 xor 11001001_2



บรรทัดที่ 7 กำหนดให้ตัวแปร val4 เก็บผลลัพธ์จากการกระทำแบบเอ็กซ์คลูซีฟออร์ระหว่างค่าในตัวแปร val3 ซึ่งมีค่าเป็น 62_{16} กับ val2 ซึ่งมีค่าเป็น $C9_{16}$ ได้ผลลัพธ์เป็น AB_{16} หรือ 10101011_2 ดังรูปที่ 4.15 ซึ่งตรงนี้จะเห็นว่าเมื่อนำค่าผลลัพธ์จากการเอ็กซ์คลูซีฟออร์มาเอกซ์คลูกซีฟออร์ด้วยค่าตัวกระทำค่าเดิมทำให้ผลลัพธ์ที่ได้เป็นค่าเดียวกับตัวถูกกระทำ

รูปที่ 4.15 ผลของ 01100010_2 xor 11001001_2

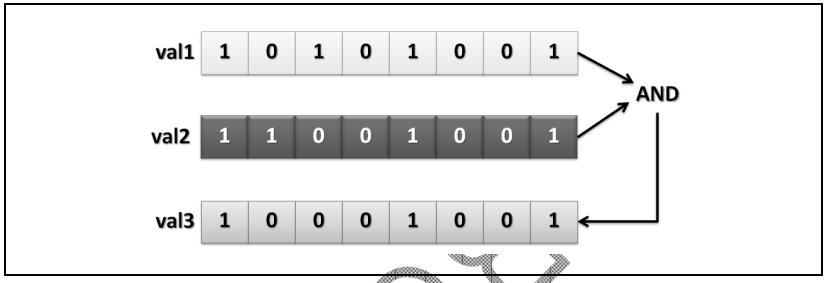


หมายเหตุ

ด้วยคุณสมบัติของเอ็กซ์คลูซีฟออร์ที่เมื่อนำมาใช้ดังบรรทัดที่ 6 และ 7 จึงมีความนิยมที่จะใช้การดำเนินการประเภทนี้กับเรื่องของการเข้าและถอดรหัสแบบง่ายๆ โดยมองว่า ตัวถูกกระทำคือข้อมูล และตัวกระทำเป็นคีย์ ซึ่งผลลัพธ์จากการกระทำเรียกว่า ข้อมูลที่ถูกเข้ารหัส เมื่อนำข้อมูลที่เข้ารหัสนี้มากระทำเอ็กซ์คลูกซีฟออร์ด้วยคีย์อีกครั้ง จะได้ผลลัพธ์เป็นข้อมูลก่อนที่เข้ารหัส และเรียกการกระทำนี้ว่า การถอดรหัส นอกจากนี้ ยังมีการใช้เทคนิคการกระทำเกี่ยวกับบิตในเรื่องของการทำภาพสไปรต์ หรือภาพโปร่งแสงอีกด้วย

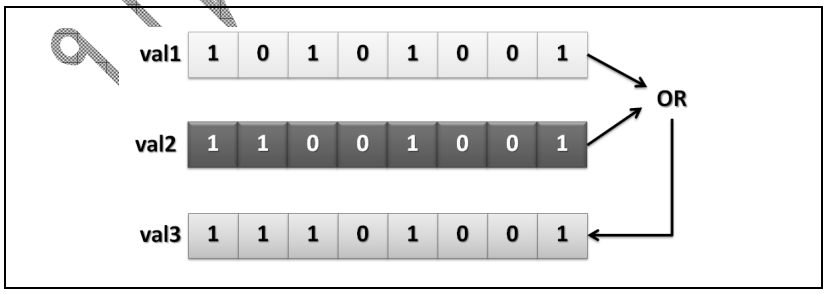
บรรทัดที่ 8 กำหนดให้ตัวแปร val3 เก็บผลลัพธ์จากการกระทำแบบแอนด์ระหว่างค่าในตัวแปร val1 ซึ่งมีค่าเป็น AB_{16} หรือ 10101001_2 กับค่าที่เก็บในตัวแปร val2 ซึ่งมีค่าเป็น $C9_{16}$ หรือ 11001001_2 ทำให้ได้ผลลัพธ์เป็น 89_{16} หรือ 10001001_2 ดัง รูปที่ 4.16

รูปที่ 4.16 ผลของ 10101001_2 and 11001001_2



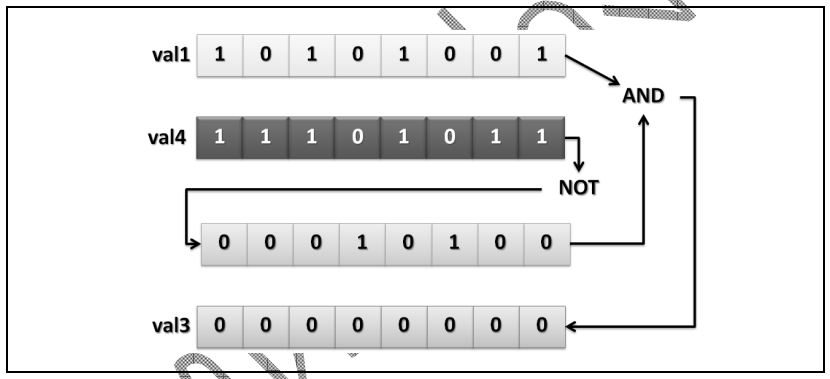
บรรทัดที่ 9 กำหนดให้ตัวแปร val4 เก็บผลลัพธ์จากการกระทำแบบออร์ระหว่างค่าที่เก็บในตัวแปร val1 ซึ่งเป็น AB_{16} หรือ 10101001_2 กับค่าที่เก็บในตัวแปร val2 ซึ่งมีค่าเป็น $C9_{16}$ หรือ 11001001_2 ทำให้ได้ผลลัพธ์เป็น EB_{16} หรือ 11101011_2 ดังรูปที่ 4.17

รูปที่ 4.17 ผลของ 10101001_2 or 11001001_2



บรรทัดที่ 10 กำหนดให้ตัวแปร val3 เก็บผลลัพธ์ของการกระทำแบบแอนด์ระหว่างค่าที่เก็บในตัวแปร val1 ซึ่งมีค่าเป็น AB_{16} หรือ 10101001_2 กับผลของการนอตของค่าที่เก็บในตัวแปร val4 ซึ่งมีค่าเป็น EB_{16} หรือ 11101011_2 ด้วยเหตุนี้ในการดำเนินงานจะต้องหาค่านอตของ val4 ก่อน เนื่องจากมีค่าความสำคัญสูงกว่า จึงได้ผลของการนอตเป็น 14_{16} หรือ 00010100_2 หลังจากนั้นเมื่อนำไปแอนด์กันจะได้ผลลัพธ์เป็น 00_{16} หรือ 00000000_2 ดังรูปที่ 4.18

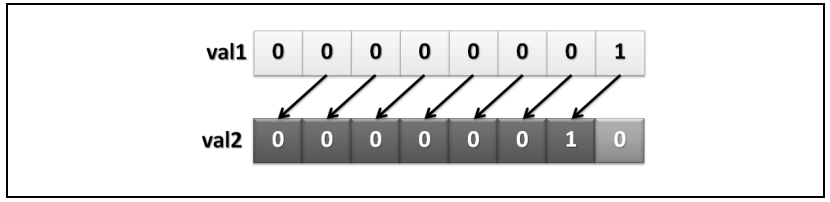
รูปที่ 4.18 ผลของ 10101001_2 and not 11101011_2



บรรทัดที่ 11 กำหนดให้ตัวแปร val1 มีค่าเป็น 01_{16} หรือ 00000001_2

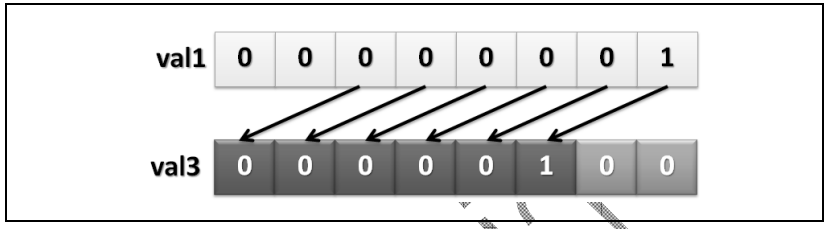
บรรทัดที่ 12 กำหนดให้ตัวแปร val2 เก็บผลลัพธ์ของการเลื่อนบิตของ val1 ซึ่งมีค่าเป็น 01_{16} หรือ 00000001_2 ไปทางซ้าย 1 บิต จึงทำให้ val2 มีค่าเป็น 02_{16} หรือ 00000010_2 ดังรูปที่ 4.19

รูปที่ 4.19 ผลของ 00000001_2 << 1



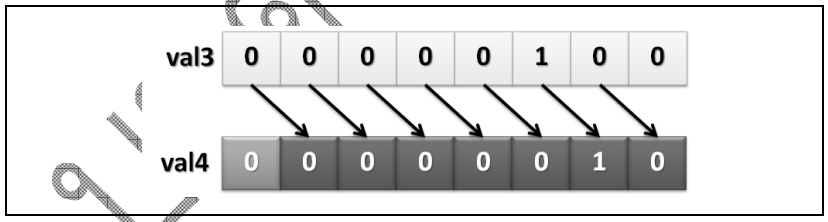
บรรทัดที่ 13 กำหนดให้ตัวแปร val3 เก็บผลลัพธ์ของการเลื่อนบิตของ val1 ซึ่งมีค่าเป็น 01_{16} หรือ 00000001_2 ไปทางซ้าย 2 บิต จึงทำให้ val3 มีค่าเป็น 04_{16} หรือ 00000100_2 ดังรูปที่ 4.20

รูปที่ 4.20 ผลของ 00000001_2 << 2



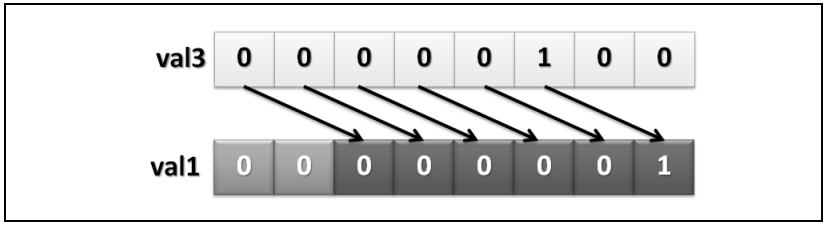
บรรทัดที่ 14 กำหนดให้ตัวแปร val4 เก็บผลลัพธ์ของการเลื่อนบิตของ val3 ซึ่งมีค่าเป็น 04_{16} หรือ 00000100_2 ไปทางขวา 1 บิต จึงทำให้ val4 มีค่าเป็น 02_{16} หรือ 00000010_2 ดังรูปที่ 4.21

รูปที่ 4.21 ผลของ 00000100_2 >> 1



บรรทัดที่ 15 กำหนดให้ตัวแปร val1 เก็บผลลัพธ์ของการเลื่อนบิตของ val3 ซึ่งมีค่าเป็น 04_{16} หรือ 00000100_2 ไปทางขวา 1 บิต จึงทำให้ val1 มีค่าเป็น 01_{16} หรือ 00000001_2 ดังรูปที่ 4.22

รูปที่ 4.22 ผลของ 00000100_2 >> 2



ดังนั้น เมื่อโปรแกรมตัวอย่าง 4.20 ทำงานเสร็จจะทำให้ตัวแปรต่างๆ มีค่าดังต่อไปนี้

val1 มีค่าเป็น 01_{16}

val2 มีค่าเป็น 02_{16}

val3 มีค่าเป็น 04_{16}

val4 มีค่าเป็น 02_{16}

หมายเหตุ

จากบรรทัดที่ 12 ถึง 15 สรุปประเด็น 2 ประเด็นดังนี้

1. เมื่อเลื่อนบิตเลขจำนวนเต็มไปทางซ้าย n บิต จะมีความหมายเดียวกันกับการคูณเลขจำนวนเต็มนั้นด้วย 2^n
2. เมื่อเลื่อนบิตเลขจำนวนเต็มไปทางขวา n บิต จะมีความหมายเดียวกันกับการหารเลขจำนวนเต็มนั้นด้วย 2^n

20.5 เครื่องหมายดำเนินการด้านตรรกะ

เครื่องหมายดำเนินการด้านตรรกะ (Boolean Operators) คือ เครื่องหมายดำเนินการที่ทำหน้าที่เปรียบเทียบผลของเงื่อนไข 1 หรือ 2 เงื่อนไข ว่าเมื่อเป็นจริงหรือเท็จ

หลักการทำงานของเครื่องหมายดำเนินการนี้มีความแตกต่างจากเครื่องหมายดำเนินการเกี่ยวกับบิตตรงที่ เป็นการเปรียบเทียบทางตรรกะ โดยมิได้พิจารณาข้อมูลในระดับบิต นั่นหมายความว่า ถ้าค่าของนิพจน์เป็น 0 หรือค่าว่าง มีความหมายเป็นเท็จในเชิงตรรกะ และถ้าค่าไม่ใช่ 0 หรือค่าว่างจะมีความหมายเป็นจริงในเชิงตรรกะ และที่สำคัญ เครื่องหมายประเภทนี้จะใช้

กับชุดคำสั่งเงื่อนไข และการวนรอบอย่างมีเงื่อนไข ซึ่งจะกล่าวถึงในบทการเขียนโปรแกรมแบบโครงสร้าง

ตารางที่ 4.11 เครื่องหมายดำเนินการด้านตรรกะ

เครื่องหมาย	ความหมาย
And	กระทำตรรกะแบบ and
Or	กระทำตรรกะแบบ or
Xor	กระทำตรรกะแบบ xor
Not	กระทำตรรกะแบบ Not

การกระทำเชิงตรรกแบบแอนด์มีรูปแบบของผลลัพธ์ดังตารางที่ 4.12 ซึ่งจะสังเกตเห็นว่าการที่จะได้ผลลัพธ์ของการดำเนินการเป็นค่า จริง ก็ต่อเมื่อ ค่าเชิงตรรกของตัวถูกกระทำและตัวกระทำมีค่าเป็นจริงทั้ง 2 ค่า

ตารางที่ 4.12 ค่าความจริงของการกระทำเชิงตรรกแบบการแอนด์

ตัวถูกกระทำ	ตัวกระทำ	ตัวถูกกระทำ **and** ตัวกระทำ
True	True	True
True	False	False
False	True	False
False	False	False

การกระทำเชิงตรรกแบบออร์ มีรูปแบบของผลลัพธ์ดังตารางที่ 4.13 ซึ่งถ้าค่าของตัวกระทำหรือตัวกระทำตัวใดตัวหนึ่งมีค่าเป็นจริงจะทำให้ผลการดำเนินการเป็นจริง และจะผลลัพธ์เป็นเท็จเพียง 1 กรณีคือ ค่าของตัวถูกกระทำและตัวกระทำเป็นเท็จทั้งคู่

ตารางที่ 4.13 ค่าความจริงของการกระทำเชิงตรรกแบบการออร์

ตัวถูกกระทำ	ตัวกระทำ	ตัวถูกกระทำ **or** ตัวกระทำ
True	True	True
True	False	True
False	True	True
False	False	False

การกระทำตรรกะแบบเอ็กซ์คลูซีฟออร์ มีรูปแบบของผลลัพธ์ดังตารางที่ 4.14 ซึ่งผลของการกระทำจะเป็นจริงเมื่อค่าของตัวถูกกระทำและตัวกระทำมีค่าที่แตกต่างกัน และผลลัพธ์จะเป็นเท็จเมื่อค่าของตัวถูกกระทำและตัวกระทำเหมือนกัน

ตารางที่ 4.14 ค่าความจริงของการกระทำเชิงตรรกแบบเอ็กซ์คลูซีฟออร์

ตัวถูกกระทำ	ตัวกระทำ	ตัวถูกกระทำ **xor** ตัวกระทำ
True	True	False
True	False	True
False	True	True
False	False	False

การกระทำเชิงตรรกะแบบนอตมีรูปแบบของผลลัพธ์ดังตารางที่ 4.15 นั้นมีวิธีใช้งานเหมือนกับการกระทำบิตแบบนอตดังรูปแบบด้านล่าง โดยผลของการกระทำชนิดนี้จะให้ผลเป็นจริงเมื่อค่าในตัวกระทำเป็นเท็จ และให้ผลลัพธ์เป็นเท็จเมื่อค่าของตัวกระทำเป็นจริง

Not ตัวถูกกระทำ

ตารางที่ 4.15 ค่าความจริงของการกระทำเชิงตรรกะแบบนอต

ตัวกระทำ	**Not** ตัวกระทำ
True	False
False	True

ตัวอย่างที่ 4.21 การใช้เครื่องหมายดำเนินการเชิงตรรกะ

บรรทัด	โค้ด
1	`program sample19`
2	`dim val1, val2, val3, val4 as byte`
3	`main:`
4	`    val1 = 20`
5	`    val2 = 40`
6	`    val3 = 30`
7	`    if ((val1 >= 10) and (val1 <= 30)) then`
8	`       val4 = 1`
9	`    end if`
10	`    if ((val2 >= val1) or (val2 <= val3)) then`
11	`       if ((val3 >= val1) xor not (val3 <= val4)) then`
12	`          val4 = 0`
13	`       end if`
14	`    end if`
15	`end.`

จากตัวอย่าง 4.21 เป็นการประยุกต์ใช้เครื่องหมายดำเนินการเชิงตรรกะ ซึ่งมีรายละเอียดการทำงานของโปรแกรมดังต่อไปนี้

บรรทัดที่ 2 ประกาศตัวแปรชื่อว่า val1 val2 val3 และ val4 โดยแต่ละตัวเป็นตัวแปรชนิด byte

บรรทัดที่ 4 กำหนดให้ตัวแปร val1 มีค่าเป็น 20

บรรทัดที่ 5 กำหนดให้ตัวแปร val2 มีค่าเป็น 40

บรรทัดที่ 6 กำหนดให้ตัวแปร val3 มีค่าเป็น 30

บรรทัดที่ 7-9 เป็นการตรวจสอบเงื่อนไข 2 เงื่อนไขด้วยตัวดำเนินการแอนด์ โดยเงื่อนไขทางซ้ายเป็นการเปรียบเทียบว่าค่าในตัวแปร val1 ซึ่งมีค่าเป็น 20 นั้นมากกว่าหรือเท่ากับ 10 หรือไม่ จึงได้ผลลัพธ์ของการเปรียบเทียบเป็น **จริง** ส่วนเงื่อนไขทางด้านขวาเป็นการเปรียบเทียบว่า ค่าในตัวแปร val1 นั้นมีค่าน้อยกว่าหรือเท่ากับ 30 หรือไม่ จึงได้ผลลัพธ์จากการเปรียบเทียบเป็น **จริง** สุดท้าย ด้วยผลลัพธ์จากเงื่อนไขทั้ง 2 จึงนำมาเปรียบเทียบเชิงตรรกะด้วยแอนด์ ดังนั้น เมื่อเงื่อนไขทางซ้ายเป็นจริง และ เงื่อนไขทางขวาก็เป็นจริง จึงทำให้เงื่อนไขของการกระทำแบบแอนด์เป็นจริง ทำให้บรรทัดที่ 8 ทำงาน นั่นคือ ทำให้ค่าของตัวแปร val4 มีค่าเป็น 1

บรรทัดที่ 10-14 เป็นการตรวจสอบเงื่อนไข 2 เงื่อนไขด้วยตัวดำเนินการออร์ โดยเงื่อนไขทางซ้ายเป็นการเปรียบเทียบว่าค่าในตัวแปร val2ซึ่งมีค่าเป็น 40 นั้น

มากกว่าหรือเท่ากับค่าที่เก็บในตัวแปร val1 ซึ่งเก็บค่า 20 หรือไม่ จึงได้ผลลัพธ์ของการเปรียบเทียบของเงื่อนไขทางซ้ายเป็น **จริง** ส่วนเงื่อนไขทางขวาเป็นการเปรียบเทียบค่าที่เก็บในตัวแปร val2ว่าน้อยกว่าค่าในตัวแปร val3 ซึ่งเก็บค่า 30 จึงทำให้เงื่อนไขเป็น เท็จ และเมื่อนำผลลัพธ์ที่เป็นจริงมาออร์กับผลลัพธ์ที่เป็นเท็จจึงทำให้การเปรียบเทียบได้ผลเป็นจริง ดังนั้น เมื่อเงื่อนไขนี้เป็นจริง จึงทำให้มีการทำงานในบรรทัดที่ 11 ถึง 13

บรรทัดที่ 11-13 เมื่อเงื่อนไขในบรรทัดที่ 10 เป็นจริง จะทำการเปรียบเทียบเงื่อนไข 2 เงื่อนไขว่าเป็นจริงหรือไม่ด้วยเครื่องหมายดำเนินการเอ็กซ์คลูซีฟออร์ โดยเงื่อนไขทางซ้ายเป็นการเปรียบเทียบค่าในตัวแปร val2 ซึ่งเก็บค่า 40 นั้นมากกว่าหรือเท่ากับค่าที่เก็บในตัวแปร val1 ซึ่งมีค่าเป็น 20 หรือไม่ จึงได้ผลลัพธ์เป็น **จริง** ส่วนเงื่อนไขทางขวาเป็นการหาน้อตของการเปรียบเทียบค่าที่เก็บใน val3 ซึ่งมีค่าเป็น 30 นั้นน้อยกว่าค่าที่เก็บในตัวแปร val4 ซึ่งมีค่าเป็น 1 หรือไม่ จึงได้ผลลัพธ์ของการเปรียบเทียบด้วยเครื่องหมายน้อยกว่าหรือเท่ากับเป็น เท็จ แต่เมื่อนำมาน้อต จึงทำให้ผลการเปรียบเทียบทางด้านขวาเป็นเท็จ ดังนั้น เมื่อผลลัพธ์ทางด้านซ้ายมีค่าเป็นจริง และผลลัพธ์ทางด้านขวามีค่าเป็นจริง เมื่อกระทำด้วยเอ็กซ์คลูซีฟออร์ จึงได้ผลเป็นเท็จ ทำให้โปรแกรมไม่ทำการประมวลผลบรรทัดที่ 12

ดังนั้น เมื่อโปรแกรมตัวอย่าง 4.21 ทำงานเสร็จสิ้นจะได้ค่าที่เก็บในตัวแปรแต่ละตัวเป็นดังนี้

val1 มีค่าเป็น 20
val2 มีค่าเป็น 40
val3 มีค่าเป็น 30
val4 มีค่าเป็น 1

21. สรุป

จากบทนี้ได้ศึกษาหลักภาษาเบสิกเกี่ยวกับโครงสร้างของโปรแกรม การเขียนคำอธิบาย สัญพจน์ คำสำคัญที่ห้ามนำไปใช้ตั้งชื่อตัวแปร/ค่าคงที่/โปรแกรมย่อย ประเภทตัวแปรที่ภาษาเบสิกของ mikroBasic PIC สนับสนุน การประกาศตัวแปร การใช้งานตัวแปร การใช้ค่าคงที่เพื่อเพิ่มความ

ยืดหยุ่นแก่โปรแกรมที่เขียน การประกาศเลเบลเพื่อใช้เป็นชื่ออ้างอิงตำแหน่งของถ้อยแถลง การใช้ตัวแปรแถวลำดับเพื่อลดความซับซ้อนในการเขียนโปรแกรม รวมถึงการประยุกต์ใช้ตัวแปรแบบแถวลำดับ สายอักขระ ตัวชี้ที่ทำหน้าที่ชี้ไปยังตำแหน่งของหน่วยความจำ การใช้โครงสร้างเพื่อจัดหมวดหมู่ข้อมูล ทำให้มองข้อมูลเป็นกลุ่มก้อน ง่ายต่อการบริหารจัดการ การสร้างชนิดตัวแปรขึ้นใช้เอง ไดเร็กทีฟควบคุมการทำงานของตัวแปลภาษา นิพจน์ และเครื่องหมายดำเนินการ เกี่ยวกับคณิตศาสตร์ การเปรียบเทียบ บิต ตรรกะ ซึ่งเป็นประโยชน์ต่อการพัฒนาทักษะการเขียนโปรแกรมภาษาเบสิก

สำหรับเนื้อหาบทที่ 5 เกี่ยวข้องกับการเขียนโปรแกรมแบบโครงสร้าง คือ เขียนโปรแกรมแบบมีลำดับขั้น เขียนโปรแกรมแบบมีเงื่อนไข เขียนโปรแกรมแบบทำซ้ำ และการเขียนโปรแกรมย่อยทั้งที่เป็นโปรแกรมย่อยแบบไม่ต้องคืนค่ากลับ และการเขียนโปรแกรมย่อยประเภทฟังก์ชันที่คืนค่าการทำงานกลับมาให้กับระบบ