บทที่ 5
แนวความคิดต่าง ๆ เกี่ยวกับการสร้างโปรแกรม
(PROGRAMMING CONCEPTS)

ก่อนที่ท่านจะเขียนโปรแกรมแรกของท่าน ท่านควรจะคุ้นเคยกับโครงสร้างพื้นฐานต่าง ๆ ณ ระดับที่ต่ำที่สุด ท่านจะมีสิ่งต่าง ๆ แต่ละสิ่งอย่างเช่น เซ็ทของอักขระ (character set) ค่าคงที่และตัวแปรต่าง ๆ จากลักษณะพื้นฐานที่สำคัญต่าง ๆ เหล่านี้ ท่านจะขยายแนวความคิดของค่าคงที่และตัวแปรต่าง ๆ เพื่อที่จะรวมเอาอะเรย์ต่าง ๆ และระดับความแม่นยำ (precisions) และชนิด (types) เข้าไว้ด้วย ท้ายที่สุด ท่านจะได้เห็นวิธีที่แนวความคิดของนิพจน์ (expressions) ต่าง ๆ จะรวบรวมทุก ๆ สิ่งของระดับต่าง ๆ มาใช้งานร่วมกัน

เซ็ทของอักขระ (Character Set)

เซ็ทของอักขระของ GWBASIC ประกอบด้วยอักขระในรูปตัวอักษร (alphabetic characters) อักขระในรูปตัวเลข (numeric characters) และอักขระพิเศษ (special characters) ต่าง ๆ

อักขระในรูปตัวอักษรประกอบด้วยอักขระในรูปตัวอักษรพิมพ์ใหญ่ (upper case letters) และอักขระในรูปตัวอักษรพิมพ์เล็ก (lower case letters) อักขระต่าง ๆ ที่เป็นตัวเลขคือเลข 0 ถึงเลข 9

อักขระต่าง ๆ ทั้งหมดที่ยังคงเหลืออยู่บนแป้นคีย์จะเป็นอักขระพิเศษต่าง ๆ อักขระบางตัวจะมีความหมายที่พิเศษเฉพาะไปยัง GWBASIC ตัวอย่างเช่น อักขระ "-" สามารถเป็นเพียงอักขระตามที่ท่านอาจจะใช้มันบนเครื่องพิมพ์ดีด (typewriter) เท่านั้น ในทางตรงข้าม ในคำสั่งของโปรแกรมของ GWBASIC อักขระตัวเดียวกันสามารถเป็นคำสั่งเพื่อที่จะทำการลบ

อักขระพิเศาต่าง ๆ บางตัวถูกสร้างขึ้นโดยการกดคีย์ 2 คีย์พร้อมกัน เซ็ทที่เห็นได้ชัดซึ่งสามารถได้มาโดยการกดคีย์ SHIFT เช่น ":" และ "?" เซ็ทอื่น ๆ ประกอบด้วยอักขระต่าง ๆ ที่ถูกสร้างขึ้นมาร่วมกับคีย์ CTRL (control) โดยทั่วไป อักขระควบคุมต่าง ๆ จะไม่ได้ทำให้

อักขระต่าง ๆ เหล่านั้นแต่ละตัวปรากฏบนจอภาพ แม้ว่าอักขระทั้งหลายมีผลต่าง ๆ ที่เป็นที่น่าสังเกตเห็นได้อย่างมากบนจอภาพก็ตาม ตัวอย่างเช่น CTRL และ L จะเคลียร์จอภาพ เซ็ทสุดท้ายของอักขระพิเศษต่าง ๆ ประกอบด้วยอักขระต่าง ๆ ซึ่งสามารถถูกสร้างขึ้นโดยตรงบนจอภาพโดยการกด ALT และทำการป้อนตัวเลขในช่วงจาก 128 ถึง 255 เข้าไป เมื่อคีย์ ALT ถูกปล่อย อักขระจะปรากฏขึ้น รายการของเซ็ทของอักขระที่สมบูรณ์สามารถถูกพบในภาคผนวก ค, ASCII Character Codes อักขระต่าง ๆ เหล่านั้นพร้อมด้วยความหมายที่พิเศษเฉพาะถูกอธิบายในตารางต่าง ๆ ต่อไปนี้

ตารางที่ 5.1

SPECIAL CHARACTERS

Character	Explanation
=	Equal sign or assignment symbol
+	Plus sign
-	Minus sign
*	Multiplication symbol
/	Division symbol
\	Integer division symbol
^	**Exponentiation** symbol
%	Declaration character for integer variables
#	Declaration **character** for double precision variables

ตารางที่ 5.1 (ต่อ)

Character	Explanation
$	**Declaration** character for string
!	Declaration character for single precision variables
,	(Apostrophe) remark delimiter
	Program statement separator
?	Abbreviation for PRINT statement
"	String delimiter

อักขระต่าง ๆ ต่อไปนี้ส่วนมากอาจจะถูกสร้างขึ้นในวิธีมากกว่า 1 วิธี

ตารางที่ 5.2

OTHER SPECIAL CHARACTERS

Key	CTRL	ALT	CHR$	Description
	C BREAK			Stops program execution and returns to GWBASIC command level.
	G	007	7	Generate a beep sound.' (B E L L)

ตารางที่ 5.2 (ต่อ)

Key	CTRL	ALT	CHR$	Description
BACKSPACE	H	008	a	Delete character to the left of cursor and move cursor to that position.
TAB	I	009	9	Move cursor 8 characters to right.
	S			Suspend program execution.
	NUM LOCK			
(most)	except:			Resume program execution.
	S			
	NUM LOCK			
ESC	U		21	Erase entire logical line.
	[			
	->			Move cursor to start of
	N			next word.
	<-			Move cursor to start of
	B			previous word.
	L	012	12	Clear screen.
	HOME			
	E			Erase to end of logical
	END			line.

ตารางที่ 5.2 (ต่อ)

Key	CTRL	ALT	CHR$	Description
INSERT	R			Toggle INSERT mode.
RETURN	M	013	13	Carriage return (end of
ENTER				logical line).

ท้ายที่สุด มีเซ็ทของคำคีย์ต่าง ๆ ซึ่งถูกสร้างขึ้นโดยคีย์ ALT ต่อไปนี้เป็นวิธีที่รวบรัดที่จะอ้างอิงไปยังคำสั่ง (statements) คำสั่ง (commands) หรือฟังก์ชันต่าง ๆ ใน GWBASIC ที่แน่นอน ตารางที่ 5.3 แสดงคำคีย์ต่าง ๆ เหล่านี้

ตารางที่ 5.3

ALT KEY WORDS

Alt	Key	Meaning	Alt	Key	Meaning	Alt	Key	Meaning
A		AUTO	G		GOTO	M		MID$
B		BSAVE	H		HEX$	N		NEXT
C		COLOR	I		INPUT	0		OPEN
D		DELETE	J		(no word)	P		PRINT
E		ELSE	K		KEY	Q		(no word)
F		FOR	L		LOCATE	R		RUN

ตารางที่ 5.3 (ต่อ)

Alt Key	Meaning	Alt Key	Meaning	Alt Key	Meaning
S	SCREEN	V	VAL	Y	(no word)
T	THEN	w	WIDTH	Z	(no word)
U	USING	X	XOR		

ค่าคงที่ (Constants)

ค่าคงที่เป็นมูลค่าจริงที่ GWBASIC ใช้ในระหว่างการประมวลผล ค่าคงที่แบ่งออกเป็น 2 ชนิดคือ ค่าคงที่สตริงก์ (string or character constant) และค่าคงที่จำนวนเลข (numeric constant)

ค่าคงที่สตริงก์เป็นชุดลำดับ (sequence) ของอักขระต่าง ๆ ในรูป alphanumeric มีความยาวสูงสุดถึง 255 อักขระถูกคลุมอยู่ภายในเครื่องหมายคำพูด ตัวอย่างเช่น :

"HELLO"

"$25,000.00"

"Number of Employees"

ค่าคงที่จำนวนเลขเป็นจำนวนบวกหรือจำนวนลบ สังเกตว่าเครื่องหมายบวกเป็น optional สำหรับจำนวนบวก ค่าคงที่จำนวนเลขต่าง ๆ ใน GWBASIC อาจจะไม่บรรจุเครื่องหมายจุลภาค ค่าคงที่จำนวนเลขแบ่งออกเป็น 5 ชนิดคือ :

1. ค่าคงที่จำนวนเต็ม (integer constant)

ตัวเลขทั้งหมดมีค่าอยู่ระหว่าง -32768 ถึง 32767 ค่าคงที่เหล่านี้จะไม่มีจุดทศนิยม

2. ค่าคงที่ในรูป fixed-point (fixed-point constant)

เลขจำนวนจริงบวกหรือลบ นั่นคือตัวเลขต่าง ๆ ที่มีจุดทศนิยม

3. ค่าคงที่ในรูป floating-point (floating-point constant)

จำนวนบวกหรือลบที่ถูกแสดงในรูปเอ็กซ์โพเนนเชียล (คล้ายกันกับสัญลักษณ์ทางวิทยาศาสตร์) ค่าคงที่ในรูป floating point ประกอบด้วยตัวเลขจำนวนเต็มหรือตัวเลขที่มีจุดทศนิยมพร้อมด้วยเครื่องหมายที่สามารถเลือกได้(แมนทิสซา) ติดตามด้วยอักษร E และจำนวนเต็มพร้อมเครื่องหมายที่สามารถเลือกได้(เอ็กซ์โพเนนท์) ค่าคงที่ต่าง ๆ ในรูป double-precision floating point จะใช้อักษร D แทนอักษร E สำหรับข่าวสารเพิ่มเติมอ้างอิงไปยัง "Numeric Precision" ข้างล่าง (อักษร E หรือ D หมายถึง "คูณด้วย 10 ยกกำลัง")

ตัวอย่าง :

235.988E-7 = .0000235988

2359E6 = 2359000000

ช่วงสำหรับมูลค่าเหล่านี้คือ 10E-38 ถึง 10E+38 (จำนวนบวกหรือจำนวนลบ)

4. ค่าคงที่เลขฐานสิบหก (hex constant)

จำนวนเลขฐานสิบหกประกอบด้วยเลขสูงสุดถึง 4 หลักและ &H อยู่ข้างหน้า เลขฐานสิบหกเป็นตัวเลขจาก 0 ถึง 9, A, B, C, D, E และ F

ตัวอย่าง :

&H76

&H32F

5. ค่าคงที่เลขฐานแปด (octal constant)

เลขฐานแปดประกอบด้วยเลขสูงสุดถึง 6 หลักและ &O หรือ & อยู่ข้างหน้า เลขฐานแปดเป็นตัวเลขจาก 0 ถึง 7

ตัวอย่าง :

&O347

&1234

<u>ระดับความแม่นยำของจำนวนเลข</u> (Numeric Precision)

ตัวเลขต่าง ๆ อาจจะถูกเก็บอยู่ในรูปจำนวนเต็ม จำนวนในรูป single precision หรือจำนวนในรูป double precision ค่าคงที่ต่าง ๆ ที่ถูกป้อนเข้าไปในรูปจำนวนเต็ม จำนวนเลขในรูปฐานสิบหก หรือจำนวนเลขในรูปฐานแปดถูกเก็บในหน่วยความจำขนาด 2 ไบท์และถูกแสดง (interpreted) ในฐานะเป็นเลขจำนวนเต็มหรือตัวเลขต่าง ๆ ทั้งหมด ใน single precision ตัวเลขต่าง ๆ สูงสุดถึง 7 หลักอาจจะถูกเก็บและถูกพิมพ์แต่เฉพาะ 6 หลักเท่านั้นจะมีความถูกต้องแม่นยำ (accurate) ใน double precision ตัวเลขต่าง ๆ จะถูกเก็บด้วยความแม่นยำของเลข 16 หลักและเลขทั้ง 16 หลักอาจจะถูกพิมพ์

ค่าคงที่ในรูป single-precision เป็นค่าคงที่จำนวนเลขใด ๆ ซึ่งไม่ใช่ค่าคงที่จำนวนเต็มและมีลักษณะใดลักษณะหนึ่งในลักษณะต่าง ๆ ดังต่อไปนี้ :

1. เป็นเลข 7 หลักหรือน้อยกว่า
2. รูปแบบเอ็กซ์โพเนนเชียลซึ่งใช้ E
3. มีเครื่องหมาย ! อยู่ข้างท้าย

ค่าคงที่ในรูป double-precision เป็นค่าคงที่จำนวนเลขใด ๆ ซึ่งมีลักษณะใดลักษณะหนึ่งในลักษณะต่าง ๆ ต่อไปนี้ :

1. เป็นเลข 8 หลักหรือมากกว่า
2. รูปแบบเอ็กซ์โพเนนเชียลซึ่งใช้ D
3. มีเครื่องหมาย # (number sign) อยู่ข้างท้าย

ตัวอย่าง :

<u>Single-Precision</u>	<u>Double-Precision</u>
46.8	345692811
-1.09E-06	-1.09432D-06

Single-Precision	Double-Precisiog
3489.0	3489.0#
22.5!	7654321.1234

ตัวแปร (Variables)

ตัวแปรต่าง ๆ เป็นชื่อต่าง ๆ ที่ถูกใช้เพื่อที่จะแทนมูลค่าต่าง ๆ ในโปรแกรมของ GWBASIC ตัวแปรต่าง ๆ แบ่งออกเป็น 2 ชนิดคือตัวแปรจำนวนเลข (numeric variable) และตัวแปรสตริงก์ (string variable) ตัวแปรจำนวนเลขโดยปกติมีมูลค่าซึ่งเป็นตัวเลข ตัวแปรอื่น ๆ ทุก ๆ ตัวประกอบด้วยสตริงก์ของอักขระต่าง ๆ ชนิด (type) ของตัวแปรต้องสอดคล้องกับชนิดของข้อมูลที่จะถูกกำหนดค่าไปยังมัน

มูลค่าของตัวแปรอาจจะถูกกำหนดขึ้นในฐานะเป็นค่าคงที่ หรืออาจจะถูกกำหนดขึ้นในฐานะเป็นผลลัพธ์ของการคำนวณต่าง ๆ หรือคำสั่งข้อมูลเข้าต่าง ๆ

ถ้าตัวแปรถูกใช้ก่อนที่มูลค่าจะถูกกำหนดไปยังมันแล้ว มูลค่าของมันจะถูกกำหนดเป็นศูนย์จนกว่าเมื่อใดจะมีการกำหนดค่า

วิธีตั้งชื่อตัวแปร (How to name a variable)

ตัวแปรของ GWBASIC อาจจะมีความยาวใด ๆ ภายในไลน์ของโปรแกรมที่มีความหมายซึ่งประกอบด้วยอักขระต่าง ๆ สูงสุดถึง 255 ตัว อักขระสูงสุดถึง 40 ตัวจะมีนัยสำคัญ (significant) ชื่อต่าง ๆ ของตัวแปรสามารถบรรจุตัวอักษร ตัวเลขและจุดทศนิยม อย่างไรก็ตาม อักขระตัวแรกต้องเป็นตัวอักษร อักขระพิเศษต่าง ๆ ข้างท้ายถูกใช้เพื่อที่จะบ่งชี้ชนิดต่าง ๆ ของตัวแปรต่าง ๆ ดู "วิธีที่จะบ่งชนิดต่าง ๆ ของตัวแปร" ข้างล่าง

ชื่อตัวแปรต้องไม่ตรงกับคำสงวน (reserved word) แต่คำสงวนต่าง ๆ ที่ถูกกำหนดอยู่ลึกเข้าไปจะถูกอนุญาตให้ใช้ได้ คำสงวนต่าง ๆ จะประกอบด้วยทุก ๆ คำสั่ง (commands) คำสั่ง (statements) ชื่อต่าง ๆ ของฟังก์ชัน และชื่อต่าง ๆ ของตัวกระทำ รายการที่สมบูรณ์ถูกจัดหาให้ในภาคผนวก ง

ถ้าตัวแปรเริ่มต้นด้วย FN แล้ว มันจะถูกกำหนดเป็นการเรียกไปยังฟังก์ชันที่ผู้ใช้บ่งบอก (user defined function) (อ้างอิงไปยังคำสั่ง DEF FN ในบทที่ 7)

วิธีที่จะประกาศชนิดต่าง ๆ ของตัวแปร (How to declare variable types)

ชื่อต่าง ๆ ของตัวแปรสตริงก์ถูกเขียนขึ้นด้วยเครื่องหมาย $ (dollar sign) ในฐานะเป็นอักขระท้ายสุด ตัวอย่างเช่น :

```
A$ = "SALES REPORT"
```

เครื่องหมาย $ ประกาศว่าตัวแปรจะแทนสตริงก์ เนื้อที่หน่วยความจำที่ต้องการ (storage requirement) จะเท่ากับ 3 ไบท์บวกด้วยความยาวของสตริงก์

ชื่อต่าง ๆ ของตัวแปรจำนวนเลขอาจจะประกาศจำนวนเต็ม จำนวนในรูป single-precision หรือจำนวนในรูป double-precision อักขระต่าง ๆ ของการประกาศชนิดสำหรับการประกาศตัวแปร จำนวนของไบท์ต่าง ๆ ที่ถูกต้องการในการที่จะเก็บชนิดของมูลค่าแต่ละชนิดจะเป็นดังต่อไปนี้ :

%	ตัวแปรจำนวนเต็ม	(2 ไบท์)
!	ตัวแปรในรูป single-precision	(4 ไบท์)
#	ตัวแปรในรูป double-precision	(8 ไบท์)

ถ้าชนิดของตัวแปรไม่ถูกประกาศแล้ว เครื่องคอมพิวเตอร์จะกำหนด default ให้เป็น single-precision จะสังเกตได้ว่า ตัวแปรต่าง ๆ ในรูป double-precision ต้องการเนื้อที่หน่วยความจำเป็น 2 เท่าของตัวแปรต่าง ๆ ในรูป single-precision ดังนั้นจึงต้องการเนื้อที่หน่วยความจำมากขึ้นสำหรับการปฏิบัติการต่าง ๆ ทางการคำนวณ

ตัวอย่างของชื่อต่าง ๆ ของตัวแปรของ GWBASIC :

PI#	double-precision
'MINIMUM!	single-precision
LIMIT%	integer

```
N$              string
ABC             single-precision
```

วิธีอื่นที่จะระบุชนิดต่าง ๆ ของตัวแปรก็คือโดยการผ่านคำสั่งต่าง ๆ ต่อไปนี้ : DEFINT, DEFSTR; DEFSNG และ DEFDBL อ้างอิงไปยังคำสั่ง DEFtype ในบทที่ 7 สำหรับข้อมูลเพิ่มเติม

ตัวแปรอะเรย์ (Array variables)

อะเรย์เป็นรายการหรือเป็นตารางเมตริกซ์ของมูลค่าจำนวนเลขหรือมูลค่าสตริงก์ต่าง ๆ อะเรย์ถูกสร้างขึ้นมาโดยการตั้ง (estiblishing) มิติ (dimensions) ต่าง ๆ สำหรับตัวแปรขึ้นมา (อ้างอิงไปยังคำสั่ง DIM ในบทที่ 7) มูลค่าแต่ละมูลค่าในอะเรย์ถูกเรียกว่าสมาชิก (element) และถูกบ่งชี้ด้วยการกำหนดระบุด้วยสับสคริพท์ (subscript) ของตัวแปร

ตัวอย่างเช่น :

```
DIM V$(4,4,2)
```

จะหมายถึงอะเรย์ 3 มิติของมูลค่าต่าง ๆ ของสตริงก์ V$ มิติต่าง ๆ อาจจะถูกคิดในฐานะเป็นแถว(rows) สดมภ์(columns) และหน้า(pages) ลำดับ (sequence) เป็นไปตามที่ผู้ใช้บ่งบอก คำสั่งนี้มูลค่าสูงสุดของสับสคริพท์ต่าง ๆ ที่ถูกสร้างขึ้นสำหรับอะเรย์ สับสคริพท์ต่าง ๆ ต้องเป็นนิพจน์ต่าง ๆ ในรูปจำนวนเต็มบวก

```
A$=V$(2,1,1)
```

กำหนดมูลค่าของสมาชิกของอะเรย์ไปยัง A$

ถ้าสมาชิกของอะเรย์ถูกสร้างขึ้นโดยปราศจากคำสั่ง DIM แล้ว อะเรย์ 1 มิติ (single dimension array) จะถูกสร้างขึ้นด้วยสับสคริพท์สูงสุดเป็น 10 โดยปริยาย

มูลค่าต่ำสุดของสัพสคริพท์จะเป็น 0 แม้ว่ามันจะถูกกำหนดเป็น 1 ก็ตาม ถ้าท่านไม่ได้ต้องการที่จะใช้สมาชิกตัวที่ 0 ในอะเรย์ ท่านสามารถประหยัดเนื้อที่ว่างของหน่วยความจำของข้อมูลโดยการใช้คำสั่ง OPTION BASE (อ้างอิงไปยังบทที่ 7) การกำหนดมูลค่าต่ำสุดเป็น 1 จะใช้เนื้อที่

8,008 ไบท์ในอะเรย์ต่อไปนี้ :

AB#(1000,2)

วิธีที่ GWBASIC แปลงค่าจำนวนเลขต่าง ๆ จากระดับความถูกต้องแม่นยำหนึ่งไปยังระดับความถูกต้องแม่นยำอื่น ๆ (How GWBASIC converts numbers from one precision to another)

กฎต่าง ๆ ต่อไปนี้จะเกี่ยวข้องเมื่อ GWBASIC แปลงค่าจำนวนเลขจากระดับความถูกต้องแม่นยำหนึ่งไปยังระดับความถูกต้องแม่นยำอื่น

1. ถ้าค่าคงที่จำนวนเลขของชนิดหนึ่งถูกกำหนดค่าไปยังตัวแปรจำนวนเลขต่างชนิดกัน ตัวเลขจะถูกเก็บตามชนิดที่ถูกระบุในชื่อของตัวแปรเป้าหมาย (target variable name)

ตัวอย่าง
```
10 A%=23.42
20 PRINT A%
RUN
 23
```

สังเกตว่า ถ้าตัวแปรสตริงก์ถูกกำหนดค่าเท่ากับมูลค่าจำนวนเลขหรือในทางกลับกัน จะเกิด "Type mismatch" error ขึ้น

2. เมื่อนิพจน์ถูกประเมินค่า ทุก ๆ ตัวถูกกระทำ (operands) ในการกระทำทางคณิตศาสตร์หรือการกระทำที่ใช้แสดงความสัมพันธ์ (arithmetic or relational operation) จะถูกแปลงค่าให้อยู่ในระดับความถูกต้องแม่นยำเดียวกัน นั่นคือ ระดับความถูกต้องแม่นยำของตัวถูกกระทำที่ถูกต้องละเอียดลออมากที่สุด (most precise operand) เช่นเดียวกัน ผลลัพธ์ของการกระทำทางคณิตศาสตร์จะถูกส่งค่าไปยังองศาของระดับความถูกต้องแม่นยำนี้

ตัวอย่าง :
```
10 D# = 6#/7
20 PRINT D#
RUN
```

.8571428571428571

การคำนวณถูกกระทำในรูปของ double precision และผลลัพธ์ถูกส่งค่าไปยัง D# ในฐานะเป็นมูลค่าในรูป double-precision

```
10 D = 6#/7
20 PRINT D
RUN
 .8571429
```

การคำนวณถูกกระทำในรูปของ double precision และผลลัพธ์ถูกส่งค่าไปยัง D (ตัวแปรในรูป single-precision) ถูกปัดเศษและถูกพิมพ์ในฐานะเป็นมูลค่าในรูป single precision

3. ตัวกระทำต่าง ๆ ทางตรรก (Logical operators) (ดูข้างล่าง) แปลงค่าตัวถูกกระทำของมันทั้งหลายไปสู่จำนวนเต็มต่าง ๆ และให้ผลลัพธ์เป็นจำนวนเต็ม ตัวถูกกระทำต่าง ๆ ต้องมีค่าอยู่ในช่วง -32768 ถึง 32767 หรือจะเกิด "Overflow" error ขึ้น

4. เมื่อมูลค่าในรูป floating-point ถูกแปลงค่าไปสู่จำนวนเต็ม ส่วนที่เป็นเศษส่วน (fractional part) จะถูกปัดเศษ (rounded)

ตัวอย่าง
```
10 C%=55.88
20 PRINT C%
RUN
 56
```

5. ระดับความถูกต้องแม่นยำไม่ได้ถูกทำให้เพิ่มขึ้นเมื่อทำการแปลงค่าจากตัวเลขที่มีระดับความถูกต้องแม่นยำต่ำไปสู่ตัวเลขที่มีระดับความถูกต้องแม่นยำสูงกว่า ตัวอย่างเช่น ถ้ามูลค่าในรูป single-precision (A) ถูกกำหนดค่าไปยังตัวแปรในรูป double-precision (B#)แล้ว เฉพาะตัวเลข 6 หลักแรกของ B# เท่านั้นจะถูกต้อง (valid) เนื่องจากว่าเฉพาะ 6 หลักของระดับความถูกต้องแม่นยำถูกจัดจ่ายให้กับ A

มูลค่าสัมบูรณ์ (absolute value) ของความแตกต่างระหว่างตัวเลขในรูป double-precision ที่ถูกพิมพ์และมูลค่าในรูป single-precision แรกเริ่มมีค่าน้อยกว่า 6.3E-8 เท่าของมูลค่าในรูป single-precision แรกเริ่ม

ตัวอย่าง
```
10 A = 2.04
20 B# = A
30 PRINT A;B#
RUN
 2.04  2.039999961853027
```

6. เมื่อทำการแปลงค่าจากมูลค่าที่มีระดับความถูกต้องแม่นยำสูงกว่าไปยังตัวแปรที่มีระดับความถูกต้องแม่นยำต่ำกว่า ผลลัพธ์ที่ได้จะถูกปัดเศษ (rounded)

ตัวอย่าง
```
10 C = 55.8834567#
20 PRINT C
RUN
 55.88346
```

สิ่งนี้ก่อให้เกิดผลต่อคำสั่งการกล่าวอ้าง (assignment statements) ต่าง ๆ เช่นเดียวกับการประเมินค่าฟังก์ชันและการประเมินค่าคำสั่ง

<u>นิพจน์และตัวกระทำ</u> (Expressions and Operators)

นิพจน์ (expression) อาจเป็นค่าคงที่สตริงก์หรือค่าคงที่จำนวนเลขหรืออาจเป็นตัวแปร เช่นเดียวกัน นิพจน์อาจจะรวมเอาค่าคงที่และตัวแปรต่าง ๆ เข้าด้วยกันกับตัวกระทำต่าง ๆ เพื่อที่จะสร้างมูลค่าเดี่ยว ๆ (single value)

ตัวกระทำต่าง ๆ กระทำการกระทำทางคณิตศาสตร์หรือการกระทำทางตรรกต่าง ๆ บนมูลค่าจำนวนเลขต่าง ๆ เช่นเดียวกันกับบนมูลค่าสตริงก์ต่าง ๆ ตัวกระทำเหล่านี้อาจจะแบ่งออกเป็นกลุ่ม (categories) ต่าง ๆ ดังต่อไปนี้ : arithmetic, relational, logical และ

functional แต่ละกลุ่มจะถูกอธิบายข้างล่าง

ตัวกระทำทางคณิตศาสตร์ (Arithmetic Operators)

ตัวกระทำทางคณิตศาสตร์ต่าง ๆ ตามลำดับก่อนหลัง (precedence) ถูกแสดงรายการในตารางที่ 5.4

ตารางที่ 5.4

Operators	Operation	Sample Expression
^	Exponentiation	X^Y
	Negation	-X
*,/	Multiplication, Floating-	X*Y
	point division	X/Y
\	Integer division	X\Y
MOD	Modulo arithmetic,	X MOD Y
+,-	Addition, Subtraction	X + Y

การหารจำนวนเต็ม (integer division) ถูกบ่งบอกโดยเครื่องหมาย \ (backslash) ตัวถูกกระทำต่าง ๆ ถูกปัดเศษให้เป็นจำนวนเต็ม (ในช่วง -32768 ถึง 32767) ก่อนที่การหารจะถูกกระทำและเศษเหลือจะถูกตัดทิ้งไปเพื่อให้เป็นเลขจำนวนเต็ม

ตัวอย่าง
```
10 A = 10\4
20 B = 25.68\6.99
30 PRINT A;B
RUN
 2  3
```

Modulo arithmetic ถูกบ่งบอกโดยตัวกระทำ MOD มันให้จำนวนเต็มที่เป็นเศษเหลือของการหารจำนวนเต็ม

ตัวอย่าง
```
10 A = 10 MOD 4
20 PRINT A
RUN
 2
```
เศษเหลือ 2 เป็นผลลัพธ์เมื่อ 10/4

```
PRINT 25.68 MOD 6.99
 5          (26/7 = 3 ด้วยเศษเหลือ = 5)
```

เพื่อที่จะเปลี่ยนแปลงลำดับในที่ซึ่งการกระทำต่าง ๆ ถูกกระทำ ให้ใช้เครื่องหมายวงเล็บ ลำดับปกติของการกระทำต่าง ๆ จะถูกดำเนินการต่อไป (maintained)

ตัวอย่างบางตัวอย่างของนิพจน์ต่าง ๆ ทางพีชคณิตและนิพจน์ต่าง ๆ ที่สอดคล้องกันของ GWBASIC ถูกกำหนดให้ในตารางที่ 5.5

ตารางที่ 5.5

SAMPLE ALGEBRAIC EXPRESSIONS AND THEIR GWBASIC COUNTERPARTS

Algrebraic Expression	GWBASIC Expression
$X+2Y$	X+2*Y
$X - \$$	X-Y/Z
$\frac{XY}{Z}$	X*Y/Z
$(X^2)^Y$	(X^2)^Y
XY^Z	X*(Y^Z)

Algrebraic Expression	GWBASIC Expression
X(-Y)	X*(-Y)

สังเกตดูในตัวอย่างสุดท้ายซึ่งตัวกระทำสองตัวที่อยู่ติด ๆ กันต้องถูกแบ่งแยกโดยเครื่องหมายวงเล็บ ถ้าในระหว่างทำการประมวลผลนิพจน์เกิดการหารด้วยศูนย์ขึ้นแล้ว ข้อความผิดพลาด (error message) "Division by zero" จะถูกพิมพ์ ค่าที่มากที่สุดของเครื่อง (machine infinity) พร้อมด้วยเครื่องหมายของ numerator จะถูกกำหนดเป็นผลลัพธ์ของการหารและการประมวลผลจะดำเนินต่อไป ถ้าการประเมินค่าของตัวกระทำการยกกำลังให้ผลลัพธ์อยู่ในรูปของศูนย์ยกกำลังด้วยจำนวนลบแล้ว ข้อความผิดพลาด "Division by zero" จะถูกแสดงผลบนจอภาพ จำนวนบวกที่มีค่ามากที่สุดของเครื่องจะถูกจัดจ่ายในฐานะเป็นผลลัพธ์ของการยกกำลังและการประมวลผลจะดำเนินต่อไป

ถ้าเกิดโอเวอร์โฟลว์ (overflow) ขึ้นในระหว่างการคำนวณแล้ว จะเกิด "Overflow" error ขึ้น ค่าที่มากที่สุดของเครื่องพร้อมด้วยเครื่องหมายที่ถูกต้องทางพีชคณิตจะถูกกำหนดในฐานะเป็นผลลัพธ์และการประมวลผลจะดำเนินต่อไป

<u>ตัวกระทำแสดงความสัมพันธ์</u> (Relational Operators)

ตัวกระทำแสดงความสัมพันธ์ (relational operators) ต่าง ๆ ถูกใช้เพื่อที่จะเปรียบเทียบมูลค่า 2 มูลค่าซึ่งอาจเป็นจำนวนเลขทั้งคู่หรือเป็นสตริงก์ทั้งคู่ ผลลัพธ์ของการเปรียบเทียบจะเป็นจริง (true (-1)) หรือไม่ก็เป็นเท็จ (false (0)) อย่างใดอย่างหนึ่ง โดยปกติผลลัพธ์นี้ถูกใช้เพื่อที่จะทำการตัดสินใจเกี่ยวกับทางเดิน (flow) ของโปรแกรม (ดูคำสั่ง IF ในบทที่ 7) ตัวกระทำแสดงความสัมพันธ์ต่าง ๆ ถูกแสดงรายการในตารางที่ 5.6

ตารางที่ 5.6

Operator	Relation Tested	Example
=	Equality	X=Y
<> or ><	Inequality	X<>Y
<	Less than	X<Y
> ,	Greater than	X>Y
<= or =<	Less than or equal to	X<=Y
		X=<Y
>= or =>	Greater than or equal to	X>=Y
		X=>Y

เครื่องหมายเท่ากับถูกใช้เพื่อที่จะกำหนดมูลค่าไปยังตัวแปร อ้างอิงไปยังคำสั่ง LET ในบทที่ 7 เมื่อตัวกระทำทางเลขคณิตและตัวกระทำแสดงความสัมพันธ์ถูกประกอบรวมกันในนิพจน์ 1 นิพจน์ โดยปกติตัวกระทำทางเลขคณิตจะกระทำก่อน ตัวอย่างเช่น นิพจน์

X+Y < (T-1)/Z

เป็นจริงถ้ามูลค่าของ X บวกกับมูลค่าของ Y มีค่าน้อยกว่ามูลค่าของ T-1 ถูกหารด้วย Z

ตัวอย่าง IF SIN(X)<0 GOTO 1000

GWBASIC ต้องประมวลผล SIN(X) ก่อนที่จะตัดสินใจว่ามันมีค่าน้อยกว่าศูนย์หรือไม่ สตริงก์ต่าง ๆ ถูกเปรียบเทียบโดยการนำเอาอักขระ 1 ตัวในแต่ละครั้งจากสตริงก์แต่ละสตริงก์ และทำการเปรียบเทียบรหัสแอสกี (ASCII codes) ต่าง ๆ ของมัน (อ้างอิงไปยังภาคผนวก ค, ASCII Character Codes สำหรับรายการที่สมบูรณ์ของรหัสต่าง ๆเหล่านี้) นานเท่านานเท่าที่รหัสแอสกีเหมือนกัน สตริงก์ต่าง ๆ จะเท่ากัน เมื่อรหัสต่าง ๆ แตกต่างกันสตริงก์ที่ประกอบด้วย

มูลค่าจำนวนเลขที่ต่ำกว่าจะอยู่ก่อนหน้าสตริงก์ที่มีมูลค่าจำนวนเลขที่สูงกว่า ถ้าในระหว่างการเปรียบเทียบจุดจบของสตริงก์หนึ่งมาถึงแล้ว สตริงก์ที่สั้นกว่าจะถูกกล่าวว่ามีมูลค่าน้อยกว่า ช่องว่างต่าง ๆ อยู่ข้างหน้าและที่อยู่ข้างท้ายจะมีความสำคัญ (significant) ตัวอย่างของนิพจน์แสดงความสัมพันธ์ต่อไปนี้เป็นจริงทั้งหมด นั่นคือ ผลลัพธ์ของการกระทำการเปรียบเทียบมีค่าเป็น -1

```
"AA" < "AB"

"FILENAME"  =  "FILENAME"

"X&" > "X#"

"CL " > "CL"

"kg" > "KG"

"SMYTH" < "SMYTHE"

B$ < "9/12/78"            (เมื่อ B$ = "8/12/78)
```

สังเกตว่าค่าคงที่สตริงก์ต่าง ๆ ทั้งหมดที่ถูกใช้ในการเปรียบเทียบนิพจน์ต่าง ๆ ต้องถูกคลุมอยู่ภายในเครื่องหมายคำพูด

ตัวกระทำทางตรรก (Logical Operators)

ตัวกระทำทางตรรก (logical operators) ต่าง ๆ กระทำการตรวจสอบต่าง ๆ บนมูลค่าจำนวนเลขต่าง ๆ การกระทำบูลีน (boolean operations) ต่าง ๆ เหล่านี้โดยปกติถูกใช้เพื่อที่จะทำการตัดสินใจโดยการเชื่อมต่อความสัมพันธ์ 2 ความสัมพันธ์หรือมากกว่าและจะส่งมูลค่าจริงหรือมูลค่าเท็จ (อ้างอิงไปยังคำสั่ง IF ในบทที่ 7 สำหรับข่าวสารเพิ่มเติม)

ผลลัพธ์ของการกระทำทางตรรก (ตารางที่ 5.7) เป็นตัวเลขซึ่งเป็นจริงถ้ามันไม่ใช่ศูนย์ หรือเป็นเท็จถ้ามีค่าเท่ากับศูนย์ ตารางต่อไปนี้แสดงรายการผลลัพธ์ต่าง ๆ ของการกระทำต่าง ๆ เหล่านี้ตามลำดับก่อนหลัง ("1" บ่งชี้มูลค่าจริงและ "0" บ่งชี้มูลค่าเท็จ)

ตารางที่ 5.7

RESULTS OF LOGICAL OPERATIONS IN GWBASIC

Operator		Results		Terminology
NOT	X	NOT X		Logical Complement
	1	0		
	0	1		
AND	X	Y	X AND Y	Conjunction
	1	1	1	
	1	0	0	
	0	1	0	
	0	0	0	
OR	X	Y	X OR Y	Disjuntion
	1	1	1	
	1	0	1	
	0	1	1	
	0	0	0	
XOR	X	Y	X XOR Y	Exclusive OR
	1	1	0	
	1	0	1	
	0	1	1	
	0	0	0	

ตารางที่ 5.7 (ต่อ)

Operator	Results			Terminology
BQV	X	Y	X BQV Y	Equivalence
	1	1	1	
	1	0	0	
	0	1	0	
	0	0	1	
IMP	X	Y	X IMP Y	Implication
	1	1	1	
	1	0	0	
	0	1	1	
	0	0	1	

ในนิพจน์หนึ่ง การกระทำทางตรรกต่าง ๆ ถูกกระทำหลังจากการกระทำทางเลขคณิตและการกระทำแสดงความสัมพันธ์ต่าง ๆ

ตัวอย่าง IF D<200 AND F<4 THEN 80

ผลลัพธ์ที่ได้จะเป็นจริงถ้ามูลค่าของ D น้อยกว่า 200 และมูลค่าของ F น้อยกว่า 4

IF I>10 OR K<0 THEN 50

ผลลัพธ์ที่ได้จะเป็นจริงถ้า I มีค่ามากกว่า 10 หรือ K มีค่าน้อยกว่า 0 หรือทั้งคู่ (I มีค่ามากกว่า 10, K มีค่าน้อยกว่า 0)

ตัวกระทำทางตรรกต่าง ๆ ทำงานโดยการแปลงค่าตัวถูกกระทำต่าง ๆ ของมันทั้งหลายไปสู่สองคอมพลีเม้นต์ของจำนวนเต็มที่ถูกกำหนดเครื่องหมาย ขนาด 16 บิทในช่วง -32768 ถึง 32767 (ถ้าตัวถูกกระทำต่าง ๆ ไม่อยู่ในช่วงนี้แล้ว จะเกิด "Overflow" error ขึ้น) การกระทำที่ถูกกำหนดให้ถูกกระทำบนจำนวนเต็มเหล่านี้บิทต่อบิทและผลลัพธ์ที่ได้ถูกตัดสินใจโดยบิทต่าง ๆ ที่สอดคล้องกันกันในตัวถูกกระทำทั้งสอง

ดังนั้น เป็นไปได้ที่จะใช้ตัวกระทำทางตรรกต่าง ๆ เพื่อที่จะตรวจสอบไบท์ต่าง ๆ สำหรับรูปแบบของบิทที่เฉพาะเจาะจง (particular bit pattern) ยกตัวอย่างเช่น ตัวกระทำ AND อาจจะถูกใช้เพื่อที่จะ "mask" ทุก ๆ บิทยกเว้น 1 บิทของบิทต่าง ๆ ของไบท์แสดงสถานะ (status byte) ที่ machine I/O port และตัวกระทำ OR อาจจะถูกใช้เพื่อที่รวมไบท์ 2 ไบท์เข้าด้วยกันเพื่อที่จะสร้างมูลค่าในรูปไบนารี (binary value) ที่เฉพาะเจาะจงขึ้นมา ตัวอย่างต่าง ๆ ต่อไปนี้จะช่วยสาธิตวิธีที่ตัวกระทำทางตรรกต่าง ๆ ทำงาน

63 AND 16 ให้ผลลัพธ์เป็น 16

เนื่องจาก 63 คือไบนารี 111111 และ 16 คือไบนารี 10000 ดังนั้น 3 AND 16 เท่ากับ 010000 ในไบนารีซึ่งมีค่าเท่ากับ 16

-1 AND 8 ให้ผลลัพธ์เป็น 8

เนื่องจาก -1 คือไบนารี 11111111 11111111 และ 8 คือไบนารี 1000 ดังนั้น -1 AND 8 เท่ากับไบนารี 00000000 00001000 หรือเท่ากับ 8

4 OR 2 ให้ผลลัพธ์เป็น 6

เนื่องจาก 4 คือไบนารี 100 และ 2 คือไบนารี 10 ดังนั้น 4 OR 2 คือไบนารี 110 ซึ่งมีค่าเท่ากับ 6

NOT X = -(X+1)

สองคอมพลีเม้นต์ของจำนวนเต็มใด ๆ จะเป็นคอมพลีเม้นต์ของบิทบวกด้วย 1

ถ้าตัวถูกกระทำทั้งสองมีค่าเท่ากับ 0 หรือไม่ก็มีค่าเท่ากับ -1 แล้ว ตัวกระทำทางตรรกจะส่งค่าเป็น 0 หรือไม่ก็เป็น -1

ฟังก์ชันจำนวนเลขต่าง ๆ (Numeric Functions)

ฟังก์ชันถูกใช้ในนิพจน์เพื่อที่จะเรียกการกระทำที่ถูกกำหนดไว้ล่วงหน้า (predetermined operation) ซึ่งจะถูกกระทำบนตัวถูกกระทำ ฟังก์ชันต่าง ๆ ที่เฉพาะเจาะจง อย่างเช่น SQR(squart root) หรือ SIN(sine) ที่มีอยู่ใน GWBASIC ฟังก์ชันเหล่านี้ถูกแสดงรายการในเนื้อหา (contents) ต่าง ๆ และถูกกำหนดรายละเอียดในหัวข้อที่เรียงลำดับตามตัวอักษรที่อยู่ต่อจากรายการ (list)

เช่นเดียวกัน GWBASIC อนุญาตให้บ่งบอกฟังก์ชันต่าง ๆ ที่เป็นของท่านเองด้วยคำสั่ง DEF FN (อ้างอิงไปยังคำสั่งนี้ในบทที่ 7)

ตัวกระทำสตริงก์ (String Operators)

ตัวกระทำสตริงก์ต่าง ๆ ประกอบด้วยฟังก์ชันการเกี่ยวโยงและฟังก์ชันสตริงก์ (concatenation and string functions) ต่าง ๆ

สตริงก์ต่าง ๆ อาจจะถูกเกี่ยวโยงหรือถูกเชื่อมต่อเข้าด้วยกันโดยการใช้ +

ตัวอย่างเช่น :

```
10 A$="FILE" : B$="NAME"
20 PRINT A$ + B$
30 PRINT "NEW " + A$ + B$
RUN
FILENAME
NEW FILENAME
```

ฟังก์ชันสตริงก์ต่าง ๆ ให้ค่าผลลัพธ์ต่าง ๆ ที่เป็นสตริงก์ต่าง ๆ ฟังก์ชันทุก ๆ ฟังก์ชันที่ถูกแสดงรายการในบทที่ 7 ซึ่งลงท้ายด้วย $ เป็นฟังก์ชันสตริงก์ โดยเพิ่มเติม คำสั่ง DEF FN สามารถถูกใช้เพื่อที่จะบ่งบอกฟังก์ชันสตริงก์เช่นเดียวกับฟังก์ชันจำนวนเลขต่าง ๆ