BASIC해설 (1)

朴 贊 震
(KIET電算室)

〈차 례〉

Ⅰ. 머 리 말

사회가 고도화되고 복잡화됨에 따라 폭주하는 정보 및 자료의 처리,생산성 향상을 위한 자동화 추구 경향은 컴퓨터의 응용 영역을 확대시켰으며 그 효율성을 더욱 증대시켰다. 국내에 컴퓨터가 도입된 이래 십 수년이 지났지만 기술 및 국산화를 위한 투자의 미흡으로 국내 컴퓨터 시장은 외국산 도입에 의존해왔던 것도 사실이다. 이에 컴퓨터의 국산화가 국내 경제발전의 필수 요소의 하나임을 인식한 정부는 제5차 과학기술 5개년 실천계획을 수립하고 반도체 및 마이크로 컴퓨터산업을 적극 지원, 86년에는 LSI 생산기술을 정착시키고 앞으로의 VLSI 기술의 기반을 구축키로 했으며 마이크로 컴퓨터의 완전 국산화와 이에 따른 자급자족을 이룩하고 앞으로의 컴퓨터 설계능력을 확보하는데 목표를 두었다. 여기에 부응해서 국내 여러 기업체에서는 컴퓨터의 개발생산에 참여하여 많은 종류의 마이크로컴퓨터를 생산해 냄으로써 마이크로컴퓨터 붐을 일으키고 있으며 이에따라 BASIC은 현대인의 필수 언어로 등장하고 있다. 지금까지 대·중·소형 컴퓨터 일변도였고 아울러 FORTRAN·COBOL·ASSEMBLER 등에 편중돼있던 컴퓨터 프로그래밍 언어도 이제 그 구각을 벗어나 BASIC으로 좀더 용이한 프로그래밍을 하는 시점에 다달았다고 하겠다. 이에 초심자의 BASIC 학습을 위해 국내에서 발간되는 컴퓨터 잡지와 국내의 마이크로컴퓨터 메이커별 BASIC 해설집을 토대로 BASIC을 설명하고자 하니 BASIC 이해에 도움이 됐으면 한다.

Ⅱ. BASIC의 탄생

컴퓨터를 일반대중이 손쉽게 사용할 수 있게 하려면 짧은 기간에 배울 수 있는 쉬운 컴퓨터 언어를 만들어 내지 않으면 안된다고 생각한 미국의 Dartmouth 대학의 교수인 J.G. Kemeny와 T.E. Kurtz 등 두사람이 여러사람의 협력을 얻어 1960년대 중반에 일반 대중을 위한 컴퓨터 언어가 탄생됐는데 이 언어가 "BASIC"이라고 불리우며 BASIC은 Beginner's Allpurpose Symbolic Instruction Code의 머리 문자를 따서 명명됐다.

컴퓨터가 알기쉬운 언어로는 기계어라고 불리우는 것이 있는데 이는 2진법의 "0"과 "1"로 쓰여진 것이다. 이것은 사람으로서는 알아보기 어려우므로 사람이 알아보기 쉬운 아셈블러 언어가 개발됐다. 이것은 기계어를 사람이 알기 쉽게 기호화한 것이다. 즉 FORTRAN, COBOL, ALGOL, PL/1 등이 그 예에 속하겠다. 그러나 이러한 언어들과 BASIC과의 큰 차이는 대화 형식인가 아닌가 하는 것이다. BASIC 언어에서는 인간과 컴퓨터가 프로그램을 1행씩 전하거나 대답을 하거나 하는 구조로 돼있다. 즉 어느 행에서 틀리면 컴퓨터는 곧 "틀렸다"고 알려준다. 이렇게 BASIC언어는 사람이 전화로 이야기 하듯이 항상 인간과 컴퓨터가 대화를 한다. BASIC이 일반에 그다지 보급되지 않은 이유의 하나로 BASIC은 초심자를 위한 언어이며 기능이 그다지 높지 않다는 인식부족을 들 수있다. 그러나 초심자라도 쉽게 이해할 수 있고 전문가 조차 부족을 느끼지 않는 기능을 가지고 있으며 항상 대화 형식으로 사용할 수 있는 BASIC은 귀중한 컴퓨터 언어라 할 수 있다.

Ⅲ. 연산자(Operator)

연산자는 어떤 주어진 값을 산술 또는 논리 연산으로 실행하며 종류는 다음과 같다.

1. 산술 연산자

4측 연산 및 지수 연산이 있으나 지수 연산은 Computer 제작회사에 따라 다

소 차이가 있으므로 생략하기로 한다.

〈 4측연산 기호표 〉

4 측연산	수 학기 호	BASIC 기호
더한다	+	+
뺀 다	−	−
곱한다	×	*
나눈다	÷	/

2. 비교 연산자

비교 연산자는 2 개의 값을 비교 하는데 쓰이며 다음의 6가지가 있다.

① A < B : A가 B보다 작다

② A > B : A가 B보다 크다

③ A = B : A와 B는 같다

④ A ≦ B : A는 B보다 작거나 같다

⑤ A ≧ B : A는 B보다 크거나 같다

⑥ A ≠ B : A와 B는 같지않다

3. 논리 연산자

논리 연산자는 빗 (Bit) 처리 또는 논리 연산을 행하는데 사용한다.

연산자	내 용
NOT	부 정
AND	논리적
OR	논리합

논리 연산자를 사용한 문의 예를 보면 다음과 같다.

AND : IF A > 10 AND B < 5 THEN·····

A가 10보다 크고 그리고 B가 5보다 작은 경우.

OR : IF C < 10 OR D > 5 THEN ·····

C가 10보다 크거나 혹은 D가 5보다 큰 경우.

NOT : IF NOT K THEN ·····

K의 값이 거짓일 경우.

Ⅳ. Commands 및 Statements

1. 문번호

다음과 같은 BASIC언어로 쓴 간단한 프로그램을 소개한다.

```
50    LET     A = 3
20    LET     B = 5
30    LET     C = A + B
40    PRINT   A,B,C
50    END
```

이 프로그램은 A = 3 , B = 5 일때, A와 B의 합을 C로하고 C의값을 계산하여 A,B,C의 값을 쓰라는 프로그램이다. LET는 A,B,C의 값이 무엇인가를 명확하게 정할때 쓰이는데 이때 주의해야 할점은 등호의 좌변에 반드시 변수를 쓰고 숫자나 등식을 써서는 안되며 등호의 우변에 수식을 쓰는 경우에는 수식에서 사용되는 변수가 그 행의 앞 어디에선가 정의돼 있어야 한다. 그러나 프로그램 문 내에서 LET는 대부분이 생략하고 사용하는 경우가 많다. 그리고 각각의 문장의 앞부분에 붙어 있는 번호를 문번호라고 부르고 있다. BASIC으로 프로그램을 쓰는 경우에 이 문 번호는 반드시 붙이지 않으면 안된다. 문번호는 0에서 65,535 ($2^{16}-1$)까지 사용할 수 있으나 BASIC의 종류에 따라서는 0에서 63999 또는 0에서 65529 까지 사용할 수 있게 되어 있다.

2. RUN (프로그램의 실행)

BASIC 언어의 프로그램을 쓰고 맨 끝에 END라고 해도 컴퓨터는 아무런 결과를 내주지 않는다. 이때의 컴퓨터는 번역된 프로그램을 정확하게 기억하고 있을 뿐이다. 결과를 원할 때는 기억하고 있는 프로그램을 실행하라는 지시를 할 필요가 있다. 그 프로그램의 실행의 지시는 RUN이라는 언어를 사용한다. 프로그램에 이상이 없으면 계산을 하고 그 결과를 알려준다. 만일 프로그램에 틀린 문장이 있으면 "? SYNTAX ERR IN 문번호"가 화면에 나타나 문번호의 내용이 틀

렸다고 알려준다.

3. PRINT

PRINT는 프로그램에서 이용한 변수의 값이나 문자를 써낼것을 컴퓨터에 지시하는 언어이다. 앞의 프로그램에서는 모두 변수 A,B,C의 값을 쓰도록 지시한 것이다. 그리고 각 변수의 중간에는 Comma가 있는 점에 주의해야 한다. Comma는 1행을 일정 간격으로 떨어져서 써내도록 지시한것인데 BASIC의 종류에 따라서 16 Column 또는 10 Column의 간격을 나타낸다. 그러나 PRINT 문안에 Semi-Colone (;)을 사용하면 수치나 문자의 공백을 비우지 않고 계속해서 써낸다. 예를들면

```
10  PRINT "A";"BCD";"EF"
20  END
]  RUN
    ABCDEF 와 같이 PRINT된다.
```

단, 경우에 따라서는 1 Column씩 공백을 나타내는 BASIC의 종류도 있다. PRINT문 안에서 프로그래머가 문자나 기호를 출력시키고 싶을 때에는 Quotation Marks (" ") 사이에 문자나 기호를 써 넣으면 된다. 즉 PRINT "문자나 기호"가 일반적인 형태이다.

4. NEW

BASIC 프로그램을 새로 입력 하고자 할때 컴퓨터내에 기억돼있는 프로그램을 모두 지우는 명령이다. 만일 프로그래머가 자신의 프로그램을 입력시키고자 할때 과거에 쓰던 프로그램이 남아 있으면 입력시키는 프로그램과 섞여 버린다. 즉 컴퓨터는 새로 입력되는 프로그램을 과거 프로그램의 수정으로 받아들이기 때문에 새로운 프로그램을 입력하고자 할 때에는 NEW를 사용해서 과거의 프로그램을 입력시켜야 한다.

5. LIST

컴퓨터에 기억돼있는 프로그램을 확인하기 위해 프로그램 전부를 써내려면 L-

IST라는 언어를 사용한다. LIST의 사용은 프로그램을 RUN시킨 결과가 이상하다고 생각되는경우 혹은 프로그램의 수정 등의 내용을 확인하기 위한 것이다.앞의 예와같이 과거의 프로그램을 지우는것을 잊었다든지 수정방법 혹은 추가방법이 틀렸을 경우에는 재빨리 잘못된 곳을 발견할 수 있는 것이다.

6. INPUT

앞에서 설명했던 LET를 이용해서 몇개의 삼각형의 넓이를 구하는 프로그램의 예를 보면 다음과 같다.

예) 삼각형 1 18 (밑변) 12 (높이)
삼각형 2 37 (〃) 62 (〃)
삼각형 3 26 (〃) 33 (〃)
삼각형 4 8 (〃) 6 (〃)
삼각형 5 10 (〃) 5 (〃)

```
10  LET  A1 = 18
20  LET  H1 = 12
30  LET  S1 = A1 * H1/2
40  LET  A2 = 37
50  LET  H2 = 62
60  LET  S2 = A2 * H2/2
70  LET  A3 = 26
80  LET  H3 = 33
90  LET  S3 = A3 * H3/2
100 LET  A4 = 8
110 LET  H4 = 6
120 LET  S4 = A4 * H4/2
130 LET  A5 = 10
140 LET
150 LET
160 PRINT  A1, H1, S1
170 PRINT  A2, H2, S2
```

```
180     PRINT    A3, H3, S3
190     PRINT    A4, H4, S4
200     PRINT    A5, H5, S5
210     END
] RUN
 18          12          108
 37          62          1147
 26          33          429
  8           6           24
 10           5           25
```

이와같이 출력이 될 것이다. 그러나 이 프로그램을 INPUT이라는 기호를 사용해서 짧게 하는 방법이 있다.

프로그램은 다음과 같다.

```
10      INPUT    A,H
20      LET      S=A*H/2
30      PRINT    A,H,S
40      END
]  RUN
?
```

RUN이라고 타이프하면 컴퓨터는 ? 를 써내고 답을 기다린다. 이 ? 는 컴퓨터가 A와H의 값이 얼마냐고 물어보는 것이다. 여기에서는 변수의 값이 A와H를 물어보므로 컴퓨터가 알 수 있도록 Comma (,)를 넣어 구획해 준다. 위 프로그램 예에서 알 수 있듯이 INPUT 기호를 사용할 때의 일반형은 다음과 같다.

```
INPUT     변수, 변수, ……, 변수
  ⋮
  ?       정수, 정수, ……, 정수
```

와 같이 INPUT 기호는 언제나 세트로 되어 있다.

7. READ와 DATA

변수의 값을 정하는 방법으로 LET와 INPUT기호의 사용을 설명했는데 제3

의 방법으로 READ와 DATA 기호의 조합이 있다. 이는 READ 기호 다음에 변수를 쓰고 그 값을 DATA기호 다음에 쓰는 것이다. 다음 2 개의 프로그램은 같은 내용의 프로그램들이다.

프로그램 1 .

```
10    LET     A = 5
20    LET     B = 20
30    LET     C = A+B
40    PRINT   A,B,C
50    END
```

프로그램 2 .

```
10    READ    A,B
20    LET     C=A+B
30    PRINT   A,B,C
40    DATA    5,20
```

READ와 DATA는 언제나 한조가 된다. (A = 5 , B = 20)

READ와 DATA의 일반형은 다음과 같다.

```
READ    변수, 변수 , ……, 변수
  ⋮
DATA    정수, 정수 , ……, 정수
```

8 . GOTO

프로그램을 작성하려면 BASIC 문장을 1 행씩 쓰고 각 행에서 번호를 쓰지 않으면 안된다. 문 번호는 크기의 순으로 붙이고 언제나 그 값이 커지도록 해야만 한다. 그러나 반복 계산에 있어서는 문번호가 작은 문으로부터 처리하는 경우에 불편한 경우가 있다. 그래서 지정한 문 번호로 점프 시키는 지시를 하는 기초가 필요하게 된다. 이것이 GOTO이다.

GOTO 기호의 일반형은

GOTO 문번호

가 되고 무조건 GOTO기호의 다음에 지정된 문 번호로 점프한다.

GOTO 기호의 사용법의 이해를 돕기위해 5개의 3각형의 면적을 구하는 프로그램을 작성해 보자.

번 호	밑변(A)	높이(H)
1	18	12
2	37	62
3	26	33
4	8	6
5	92	74

```
10      READ      A,H
20      LET       S=A*H/2
30      PRINT     A,H,S
40      GOTO      10
50      DATA      18,12,37,62,26,33,8,6,92,74
60      END
] RUN
 18          12          108
 37          62          1147
 26          33          429
 8           6           24
 92          74          3404
```

GOTO 기호를 사용하지 않으면 프로그램이 길어진다는 것은 LET 나 INPUT 의 예를 비교해보면 알수 있을 것이다. 또 앞의 예에서 행번호 40 에서 행번호 10 으로 행번호가 작은 쪽으로 점프했으나 행번호가 큰 쪽으로 점프하는 경우에도 GOTO 기호가 사용된다.

9. IF …… THEN ……

무조건 점프의 지시를 하는경우 GOTO 기호를 사용하면 된다는 것은 이해됐을 것으로 생각되므로 여기서는 조건부 점프의 지시 방법에 대해 설명한다.

조건부 점프의 몇가지 예를 들어보면

```
IF  A=O          THEN  문번호
IF  A>O          THEN  문번호
IF  A<O          THEN  문번호
IF  A=B+C        THEN  문번호
```

IF A＊B＞C THEN 문번호

IF A－B≦C＊3 THEN 행번호

예의 내용을 살펴보면 A＝O, A＞O, A＜O, A＝B＋C, A＊B＞C, A－B≦C＊3등을 조건이라고 한다. 즉 이들 조건이 만족되는 경우에만 지정된 문 번호로 점프하는것이 조건부 점프이다. 그리고 BASIC에서 IF……THEN……의 일반형은 다음과 같다. IF 조건 THEN N. N은 조건이 성립될때 점프하는 문번호이다. 만일 조건이 성립되지 않는 경우에는 IF……THEN…… 문장 다음의 문 번호를 실행한다. 그러면 주어진 임의의 수를 읽고 마이너스가 아닌 수를 써내는 프로그램을 작성해 보자.

```
10    READ  X
20    IF  X<O  THEN  10
30    PRINT  "X = ", X
40    GOTO  10
50    DATA  3,-1,5,8,-6,0,2,-8
60    END
] RUN
 X = 3
 X = 5
 X = 8
 X = 0
 X = 2
```

10. FOR와 NEXT

BASIC언어의 주역은 앞에서 설명한 IF……THEN……기호와 이제부터 설명하는 FOR/NEXT 기호라고 해도 과언은 아니다.

FOR/NEXT 일반형은 다음과 같다.

FOR 변수=초기치 TO 최종치 STEP 변동폭

NEXT 변동폭

여기에서 초기치 최종치 및 변동폭은 변수, 정수 또는 식의 어떤 것이라도 좋지만 그 값이 정해지는것이 아니면 안된다. 그리고 FOR와 연결해서 쓰는 NEXT 다

음의 변수는 같은 기호가 아니면 안된다. FOR 와 NEXT를 사용한 프로그램의 예를 들면 다음과 같다.

```
10    FOR     I = 1    TO     10
20    PRINT   I
30    IF      I = 5    GOTO   50
40    NEXT    I
50    END
] RUN
 1
 2
 3
 4
 5
```

FOR/NEXT 루프의 FOR 스테이트먼트에서 지정되는 초기치, 최종치, 변동폭은 정수이든 변수이든 혹은 식이라도 관계가 없다. FOR/NEXT 루프를 사용하는 경우 주의해야 할 점이 몇가지 있다.

첫째 FOR/NEXT 루프안에 또 하나의 FOR/NEXT 루프를 사용하는 것은 허용되지만 안에서 수행되는 FOR/NEXT 루프는 완전히 밖의 FOR/NEXT 루프 안에서 작업을 수행할 수 있도록 구성되어져야 한다.

둘째 FOR/NEXT 루프 안에서 밖으로 뛰어 나오는것은 허용되지만 루프 밖으로부터 FOR/NEXT 루프 안으로 뛰어드는 것은 허용되지 않는다.

11. REM

REM은 비실행문으로서 프로그램을 실행 할때는 무시된다. 프로그램의 문수가 많아지는 경우에는 군데군데 주석을 써 넣으면 작성자도 이용자도 알기 쉽다. REM 문에는 모든 영문 숫자가 포함되며 최대 길이는 다른 문에서와 마찬가지로 255 문자이다.

12. SAVE 와 LOAD

고생하여 만든 프로그램을 보존할 수 없다면 마이크로 컴퓨터는 많은 사람들에

게 외면당할 것이다.

프로그램을 보존하는 방법에는 종이테이프, 카세트테이프, 프로피디스크등 여러가지가 있으나 마이크로컴퓨터 세계에서는 카세트테이프가 현재 가장 많이 보급돼 있다고 할수 있다. 여기서는 카세트테이프에 프로그램을 기록하거나 읽어내는 방법에 대해 설명한다.

정확하게 만든 프로그램이 컴퓨터 안의 메모리에 남아있다고 하면 컴퓨터에 프로그램을 카세트테이프에 보존하라는 지시를 한다. 이 지시를 하려면 SAVE "ABCD" 하고 RETURN Key를 누르면 프로그램 이름이 ABCD로서 테이프에 담겨진다. 그러나 Altair BASIC에서는 SAVE 기호의 머리에 C를 붙여 C-SAVE라고 한다. SAVE 나 CSAVE는 똑같은 기능을 가진다. 다음에 카세트테이프에 보존하고있는 프로그램을 읽어내는 방법에 대해 알아보자. 카세트테이프에 기록돼 있는 프로그램을 읽어내라는 지시를 하려면 LOAD라는 기호를 쓴다.

Altair BASIC에서는 CLOAD를 쓴다. 프로그램을 기록할 때에 명명한 프로그램의 약호를 쿼테이션마크안에 쓰면 컴퓨터는 그 약호가 붙은 프로그램을 찾아내어 읽어내기 시작한다. SAVE 기호로 프로그램을 테이프에 기록시킬 때에는 컴퓨터안의 메모리에는 프로그램이 남아 있으나 LOAD기호로 프로그램을 카세트테이프로부터 읽어낼 때에는 컴퓨터안의 메모리에 남아 있는 프로그램은 전부 지워져 버리는데 주의 해야 한다.