連載

BASIC에 의한 컴퓨터 프로그램(6)

編 輯 室

2-2 FIELD 문

일반형은 다음과 같다.

FIELD #화일번호, 길이 AS 문자열 변수,…, 길이 AS 문자열변수

《예》 FIELD #1, 10 AS A$, 2 AS K$

이것은 "1번 화일에 관해서는 레코드 처음의 10 바이트를 A$, 이것에 이어진 2 바이트를 K$로 한다"라는 의미가 된다.

2-3 LEST문, RSET문

일반형은 다음과 같다.

LSET 필드명=문자열식

RESET 필드명=문자열식

《예》 LSET A$=NAME$

RSET K$=AGE$

와 같이 쓰고, 이것은 각각 "왼쪽부터 대입하라" "오른쪽부터 대입하라"라는 의미이다. 대입처 필드인 좌변 바이트 수보다 대입해야 할 문자열인 우변의 바이트 수가 많을 때 넘치는 부분은 무시된다. 여기서 주의하여야 할 사항은 필드에 대입은 LSET 문 또는 RSET 문에 한해서 가능하며, 일반적인 대입문에서는 기록하지 못한다.

2-4 함수 MKS $, MKI $, MKD $

LSET 문이나 RESET 문으로 필드에 대입할 수 있는 것은 문자열 데이터에한해야 하며, 수치형 데이터를 대입하려면 먼저 문자열형으로 변환해 둘 필요가 있다. 이 경우에는 함수 STR$를 사용하는 것이 하나의 방법이지만, STR$로 변환하면 2진법→ 10진법의 변환에 시간이 걸리고, 또한 바이트 수가 많아진다. 예를 들면 정수형 데이터는 2 바이트로 표현되어 있으나, STR$로 변환하며 부호와 최대 5자리의 숫자가 나열되므로 경우에 따라서는 6바이트로 된다. 프린터 등에 출력하려면, 이와 같이 그대로 변환할 필요가 있으나 디스크에 기록하는 것이 목적일 경우에는 2진→ 10진 변환 등은 필요없고, 2진법의 비트패턴(bit pattern)을 그대로(정수형이면 2바이트, 실수형이면 4바이트, 배정도 실수형이면 8바이트의 문자열로) 사용해 주면 된다. 이를 위한 함수가

MKS$(실수형 변수명 또는 식)

MKI$(정수형 변수명 또는 식)

MKD$(배정도 실수형 변수명 또는 식)

이다. 이들 함수를 사용하면 비트패턴은 그대로 바꾸어지지 않고 데이터형을 문자열형으로 변경해 준다.

《예〉 10 I%=65*256+66

20 C$=MKI$(I%)

30 PRINT C$

OK

라는 프로그램을 실행해보면 알 수 있다. 첫문을 실행하면 정수형 변수 I%의 값은 16706으로 되며, 내부에서는 2진법

01000001010000010

으로 표시되며, 2번째 문을 실행하면 이것이 2바이트 문자열

0100000 01000010

문자A를 나타낸다. 문자B를 나타낸다.

로 간주되어 문자열 변수 C$에 대입된다. 마지막 문에서는 그것을 출력하므로 문자열 BA가 표시된다.

2-5 함수 CVS, CVI, CVD

이들은 앞에서의 MKSS$, MKI$, MKD$의 역함수로 각각 4바이트, 2바이트, 8바이트의 문자열형 데이터 비트패턴을 그대로 실수형, 정수형, 배정도 실수형의 데이터로 취급하는 기능을 가진다.

2-6 PUT문과 GET문

PUT문은 버퍼의 내용을 디스크에 기록하는 명령이며, GET문은 디스크의 내용을 버퍼에 입력하는 명령이며, 일반형은 다음과 같다.

PUT # 번호, 레코드 번호
GET # 번호, 레코드 번호

《예》 PUT # 1, N
PUT # 1, 7

2-7 함수 LOC 및 LOF

LOC는 마지막에 직접처리 화일(random file)에서 읽어진 레코드 번호, 또는 화일에 기록된 레코드 번호, LOF는 화일 크기를 바이트 수로 돌려준다. 단, BASIC에 의하여 작성된 디스크 화일에 대해서는 128의 배수가 되며, 예를 들면 실제의 데이터가 310 바이트일 때 LOF 함수는 384를 돌려준다. 또한 인수는 화일의 OPEN시에 사용된 화일번호이다.

기타 명령과 함수

1. READ 문과 DATA 문

이제까지의 프로그램 예에서는 데이터의 입력방법으로서,

· 입력지시를 INPUT문으로 사용하고
· 실제 데이터는 실행시점에서 입력

하는 방법을 취급해 왔는데, 또 하나의 방법으로서,

· 데이터를 프로그램과 함께 넣어두고
· 그 입력지시를 READ 문으로

하는 방법이 가능하다. 즉, 입력할 데이터를

· DATA 첫번째 데이터, 2번째 데이터,…

의 형식으로 프로그램 중에 기입하면, 그것을 READ 문으로 읽는다. READ 문의 사용법은 INPUT 문과 같은 형식

READ 변수명, 변수명,…

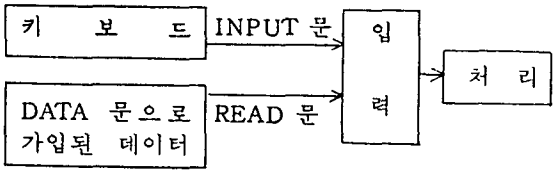
이다. INPUT 문과 READ 문의 차이점은

· INPUT 문은 실행시에 키보드에서 데이터를 읽고
· READ 문은 DATA 문으로 기입한 데이터를 읽는다.

라는 점이 있다. DATA는 READ 문의 조합에 의한 방법에는 다음과 같은 장점이 있다.

(1) 통계 계산 등에서 같은 데이터를 여러가지 수법으로 해석하고 싶을 경우에 같은 데이터를 몇번이나 반복하여 입력하지 않고 처리한다.
(2) 확인(debug)할 때 실행 데이터를 한번만 입력하면 된다.
(3) RESTORE 문을 써서 "현재 데이터의 최초 위치에 되돌아가라"라는 지시가 가능하므로 하나의 프로그램 내에서 같은 데이터를 몇번이라도 읽을 수가 있다.

INPUT 문으로 입력한 데이터는 바로 지워지고 DATA 문으로 기입한 데이터는 프로그램의 일부로 기억장치에 남아 있으므로 같은 데이터를 계속 이용할 수 있다.



READ 문의 일반형
READ 변수형, 변수명,…
DATA 문의 일반형
DATA 값, 값,…
DATA 문은 프로그램 내에서 어느 곳에 있어도 된다.
RESTORE 문의 일반형
RESTORE
또는
RESTORE 입력시작 행번호

"입력시작 행번호"를 지정하지 않으면 최초의 DATA 문에서 순서대로 입력되고, 지정하면 지정한 행의 DATA 문에서 입력한다. 이 기능에 의하여 여러가지 경우의 데이터를 확실하게 구별할 수 있다.

《예》 [프로그램1] 평균치와 표준편차

```
 110 REM··· 평균치와 표준편차
 120 READ N
 130 W=0
 140 FOR I=1 TO N
150     READ X : W=W+X
 160 NEXT I
 170 M=W / N
180 RESTORE 260
 190 W=0
  200 FOR I=1 TO N
  210     READ X : W+ (X−M)^2
  220 NEXT I
  230 S=SQR (W / N)
  240 PRINT "평균치" ; M, "표준편차" ; S
  250 DATA 18
  260 DATA 23, 48, 51, 82, 66, 58
  270 DATA 73, 71, 56, 18, 35, 68
  280 DATA 57, 63, 80, 75, 61, 52
   run
   평균치 57.61111
   표준편차 17.45196
```

2. INPUT $

INPUT$는 키보드에서 지정된 문자수만 입력하기 위한 기능이며, 문법상 함수로 취급되고, 인수의 위치에 지정할 문자수를 쓴다.

일반형은

문자열 변수명=INPUT$(문자수)

《예》 A$=INPUT$(6)

INPUT 문과의 차이점은

(1) 명령이 아니고 함수이다.

(2) 문자수 지정 방식이다.

(3) 입력 문자열의 끝에 리턴[RETURN] 키를 누를 필요가 없다.

(4) 입력요구를 나타내는 ?나 커서(cursor)가 나타나지 않는다.

(5) 입력된 문자가 화면에 나타나지 않는다.

(6) 문자 이외의 키, 예를 들면 [RETURN] [ESC] [BKSP] 등도 ASCII 코드의 형태로 입력할 수 있다.

(7) 키를 잘못 눌렀을 때 커서를 되돌려서 정정할 수 없다.

사용 예를 들어 보면 다음과 같다.

키보드에서 1자 입력하면, 그 ASCII 코드를 나타내는 프로그램을 작성하여 보자.

```
 10 C$=INPUT$(1)
 20 PRINT ASC(C$)
 30 GOTO 10
```

3. INKEY $

이것은 "키보드의 현재 상태"를 살피는 함수이며,

· 키를 누르고 있으면 그 문자가 입력되고

· 키를 누르지 않으면 공열(" ")이 입력된다.

와 같은 함수인데 인수는 필요없으며, 일반적으로

문자열 변수명=INKEY$

《예》 A$=INKEY$

와 같이 사용한다.

앞절에서 설명한 INPUT$와 같이

· [RETURN] 키를 누를 필요가 없고

· 「?」도 커서도 나타나지 않고

· 입력한 데이터가 화면에 표시되지 않고

· 문자 이외의 키를 입력할 수 있다.

등이 INPUT 문과 다르며, 그러한 의미에서는

INPUT$ (1)

과 비슷하지만, INPUT$는 키를 누를 때까지 기다리는데 반하여 INKEY$는 키가 눌러져 있지 않으면 "키나 눌러져 있지 않음"이라는 정보를 주고 다음으로 가는 점이 다르다.

4. LINE INPUT

이것은 [RETURN] 키가 눌러질 때까지를 하나의 문자열로 입력하는 명령으로 다음과 같이 사용

한다.

LINE INPUT 문자열 변수명

또는

LINE INPUT "입력 요구", 문자열 변수명

일반적인 INPUT 문과 다른 점은

· 「, 」나 「: 」이나 「 " 」를 포함한 문자열을 그대로 입력할 수 있다.
· 입력 변수명은 하나만 쓸 수 있다.

틀린 예를 들어보면 다음과 같은 것이 있다.

LINE INPUT A$, B$

· 입력 대기를 나타내는 「?」는 표시되지 않음 ("입력요구 메세지"의 뒤에 「; 」을 해도 「?」가 나오지 않는다.)

등이다. 전형적인 용도로서는 문장의 입력, 기호열(수식 등)의 입력, 프로그램의 입력 등이 있다.

4-1 LINE INPUT# 문

디스크 화일의 입력에도 LINE INPUT 문을 (#를 붙인 형으로)사용할 수 있다.

사용 예를 들어보면,

내용이나 서식을 알지 못하는 화일이 있을 때 내용을 확인할 때 보통의 INPUT 문으로는 어려우므로 다음과 같이 읽는다.(단, 순서적 처리 화일에 한함)

[프로그램2] LINE INPUT# 문

```
110 INPUT "화일명은 " ; F$
120 OPEN F$ FOR INPUT AS #1
130 WHILE NOT EOF (1)
140    LINE INPUT #1, A$
150    PRINT A$
160 WEND
170 CLOSE #1
```