BASIC에 의한 컴퓨터프로그램(5)

編 輯 室

9. 합 계

n개의 데이터(data), a_1, a_2,……, a_n을 입력하여

$S=a_1+a_2+\cdots\cdots+a_n$

을 계산해서 인쇄하는 처리 과정을 다음과 같이 생각할 수 있다.

(1) 부분합을 나타내는 변수 예를 들면 S를 설정하여, 그 값의 시작값을 0으로 한다.

(2) 제1항, 제2항, 제3항,……의 순서로 마지막 항 까지 반복하면서 각항의 값 a_i를 부분합에 더한다.

(3) 부분합의 마지막 값이 전체합계로 된다.

〈프로그램 예〉 합계(1)

```
10  INPUT  N
20  DIM  A(N)
30  FOR  I=1  TO  N
40    INPUT  A(I)
50  NEXT  I
60  S=0
70  FOR  I=1  TO  N
80    S=A(I)+S
90  NEXT  I
100  PRINT  "합계" ; S
```

다음과 같은 방법으로 해도 된다.

〈프로그램 예〉 합계(2)

```
10  INPUT  N
20  S=0
30  FOR  I=1  TO  N
40    INPUT  A
50    S=A+S
60  NEXT  I
70  PRINT "합계"  ; S
```

10. GOUSB~RETURN

이 GOSUB 문은 어떤 부분적인 처리를 한 뒤에 처음의 장소로 되돌아와서 계속하여 실행하려고 할 때에 사용되는 문으로 다음과 같이 사용한다.

여기서 SUB는 subroutine의 의미이다.

```
GOSUB  호출할 행번호
```

이 문을 실행하면, 행번호로 지정된 곳으로 뛰어서 그 곳의 프로그램을 실행하고,

```
RETURN
```

이라는 문에 도달하면, 처음의 장소, 즉 GOSUB 문의 다음 문으로 되돌아 오게 된다.

BASIC은 다른 프로그램 작성 언어인 FORTRAN이나 PASCAL과는 달라서 부 프로그램은 변수, 행번호 등이 주 프로그램과 모두 공통이며, 인수를 쓸 수 없다. 다른 언어의 부프로그램과 같은

기능이 필요한 때는 DISK BASIC의 Chain job 이라는 형태를 사용하면 된다.

GOSUB 문의 간단한 사용 예를 들어 보면 다음과 같은 것이 있겠다.

키보드에서 데이터를 입력하였을 때, 값이 올바르게 되어 있는가를 살펴서, 틀리면 경고를 출력하고 다시 입력할 것을 요구하는 처리가 실용 프로그램에서 가끔 필요하게 된다.

이와 같이 자주 사용되는 과정을 부 프로그램이라는 형태로 작성하여 두고 필요할 때마다 그것을 GOSUB 문으로 호출하여 쓰면 편리하다.

이와 같은 부 프로그램의 예는 다음과 같다.

〈프로그램 예〉

```
210  REM  "범위 확인"
220  INPUT  X
230  IF  X<0   OR  X>400  GOTO  250
240  RETURN
250  BEEP
260  PRINT  "이 값은 안맞음"
270  PRINT "0이상 400이하가 아니면 안됨"
280  PRINT "다시 입력 하시오"
290  GOTO  220
```

이 부 프로그램을 호출하여 이용하는 주 프로그램의 한 예로써, n개의 물건 중에서 r개를 취하는 조합의 수 ${}_nC_r$을

$${}_nC_r=\frac{n!}{r!\ (n-r)!}$$

라는 공식으로 계산하는 프로그램은 다음과 같다.

〈프로그램〉 조합

```
10  REM  "조합"
20  GOSUB  100
30  PRINT "n=" ; : GOSUB  210 : N=X
40  PRINT "r=" ; : GOUSB  210 : R=X
50  NCR=NK(N) / (NK(R)  * NK(N-R))
60  PRINT  "nCr=" ; NCR
70  PRINT
80  GOTO  30
```

이 프로그램의 행 20에서 부르고 있는 것은 계승의 표를 작성하는 프로그램이며, 그 내용은 다음 부 프로그램과 같다.

〈프로그램〉 계승

```
100  REM  "--N!  표작성--"
110  D/M  NK(30)
120  NK(0)=1
130  FOR  I=1  TO  30
140    NK(I)=NK(I-1) *I
150  NEXT  I
160  RETURN
```

〈실행 예〉 RUN

```
n=?  10
r=?  3
nCr=120
n=?  7
r=?  2
nCr=21
```

11. ON-GO TO, ON-GOSUB

11-1. ON~GO TO

프로그램 수행중 한번에 여러 방향으로 분기해야 할 경우가 있다.

번호를 선택의 기준이 키(key)로 해서 이와 같이 분리하려면,

ON 번호 GO TO 행선1, 행선2,……, 행선n 이라는 문을 사용하면 편리하며, 이것은

```
번호가 1이면 행선 1로 가라
번호가 2이면 행선 2로 가라
      :            :
      :            :
n이면 행선 n으로 가라
```

라는 것을 나타내고 있으며, 행선은 행번호로 지정한다.

〈예〉 ON K GOTO 100, 200, 300, 400

번호의 위치에 식을 사용해도 가능하다.

〈예〉 ON N-5 GOTO 1100, 1200, 1300, 1400

식의 값이 정수가 아닌 소수인 경우는 소수부분이 반올림 된다.

그 결과가 1−n 이외로 되었을 때, 즉 해당하는 행선이 없을 때는 다음의 문으로 간다.

11−2. ON~GOSUB

같은 형식으로,

ON 번호 GOSUB 행선1, 행선2,……, 행선n

이라는 방식으로 부 프로그램을 부르는 분기 방식이 있다. 이때는 번호에 의하여 선택된 곳으로 가서, 그곳의 프로그램을 실행하고 RETURN 문에 도달하면 그곳에서 ON 문의 다음 문으로 되돌아 온다.

이 방식의 간단한 사용 예를 살펴보면 다음과 같은 3각형의 면적을 계산하는 프로그램을 들 수 있겠다.

〈사용 예〉 ON 문은 일반 프로그램의 기능선택에 자주 사용된다. 다음에 표시한 것은 아주 간단한 예로 3각형의 면적을 계산하는 프로그램으로 공식은 다음과 같다.

(1) 공식1 $S=ab\ \sin\theta/2$

(2) 공식2 $S=ah/2$

(3) 공식3 $Z=(a+b+c)/2$

$$S=\sqrt{Z(Z-a)\ (Z-b)\ (Z-c)}$$

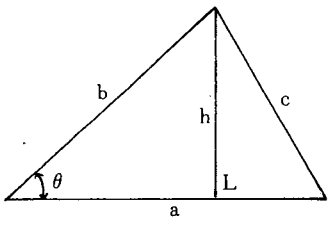

그림. 삼각형의 면적계산

삼각형의 면적을 계산하는 여러가지 조건의 하나를 키보드에서 지정하여 계산할 수 있다.

〈프로그램〉 삼각형의 면적계산

```
10  PRINT "삼각형의 면적"
20  PRINT "계산식 2"
30  PRINT "공식1 (2변과 각도)"
40  PRINT "공식2 (밑변과 높이)"
50  PRINT "공식3 (3변의 길이)"
60  PRINT "공식의 번호?"
70  PRINT "공식"      ; N
80  IF  N >3  GOTO  70
90  ON  N  GOTO  110, 150, 190
100  REM"--공식1 (2변과 각도)--"
110  INPUT"A, B, THETA="; A, B, T
120  S=A* B* SIN(T / 57.2958) / 2
130  GOTO  220
135  REM "------"
140  REM  --공식2 (밑변과 높이)--
150  INPUT "A, H="; A, H
160  S=A*H / 2
170  GOTO  220
175  REM  ------
180  REM "--공식3 (3변의 길이)--"
190  INPUT "A, B, C="; A, B, C
200  Z=(A+B+C) / 2
210  S=SQR(Z*(Z-A) *(Z-B) *(Z-C))
220  PRINT "S="; S
230  END
```

```
run
삼각형의 면적
계산식은
  공식1 (2변과 각도)
  공식2 (밑변과 높이)
  공식3 (3변의 길이)
  공식의 번호 ?
  공식      ? 3
  A, B, C=? 3, 4, 5
  S=6
  OK
```

디스크 사용법

디스크를 데이터 기억용으로 사용하려면 크게 나누어서 순서적처리(sequential access)와 직접처리(random access 또는 direct access)라는 두 가지 방식이 있다.

1. 순서적 처리 방법

순서적 처리 방법으로 디스크를 사용하려면

먼저 OPEN 문에 의하여 화일을 열어준 다음 출력은 WRITE #문으로, 입력은 INPUT #문으로 하고, 끝나면 CLOSE 문에 의하여 화일을 닫는다.

위와 같은 요령으로 프로그램을 작성하면 된다. OPEN 문의 일반형은 다음과 같다.

출력은 : OPEN 화일명 FOR OUTPUT AS # 번호

입력은 : OPEN 화일명 FOR INPUT AS # 번호

화일명을 붙이는 방법은 문자를 8자 이내로 하여 사용한다. 여기서 사용할 수 있는 문자는 영문자, 숫자와 ! # $ W % + - / ( )가 사용될 수 있다.

화일명의 앞에 원칙적으로 「장치명」을 다음과 같은 형으로 써야 한다.

윗쪽 플로피 디스크(floppy disk)는 A:

아래쪽 플로피 디스크(floppy disk)는 B:

하드 디스크(Hard disk)는 C:

다만, A: 일 때는 기입하지 않아도 된다.

끝부분의 지정부분 「# 번호」는 화일 번호 또는 채널(Channel) 번호라 하고 뒤에 INPUT# 문이나 WRITE# 문으로 그 화일을 지정하는데 사용된다. 이들 문에는 화일명을 직접 참조할 수 없다.

이 번호는 정수형으로 1에서 15까지 범위의 번호를 붙일 수 있다.

〈프로그램 LINE INPUT 문

```
10 OPEN"C:PPP.BAS"FOR INPUT AS #6
20 LINE INPUT #6, A$
30 PRINT A$
40 CLOSE #6

SAVE "PPP. bas," a
OK
run
10 OPEN"C: PPP. BAS"FOR INPUT AS #6
OK
```

1-1. 추가출력

순서적 처리에서는 화일 가운데 있는 데이터를 바꾸어 쓸 수 없고 항상 새로운 화일을 만들어 사용해야 하지만, 기존의 화일 뒤에 추가는 가능하며, 이 경우에는 화일을 다음과같이 열어주면 된다.

OPEN"화일명"FOR APPEND AS # 번호

또, 위의 방법 외에

출력은 : OPEN"O," #번호, "화일명," 레코드 길이

입력은 : OPEN"I," #번호, "화일명," 레코드 길이

와 같은 형식으로 사용할 수 있다.

INPUT #문, WRITE #문은 다음과 같이 사용한다.

INPUT #번호, 입력 항목

WRITE #번호, 출력 항목

〈예〉

```
INPUT  #1,  A,  B$,  C(I)
WRITE  #2,  I,  J$,  L
```

WRITE #문 대신 PRINT #문을 사용할 수도 있다.

이것은,

PRINT #번호, 출력 항목

과 같은 형식으로 쓰며, 문법은 PRINT 문과 거의 같은 형식이다. 그러나 WRITE # 문을 사용하면 다음과 같은 장점이 있다.

① 출력 항목의 구분에 자동적으로 컴마가 삽입되고

② 문자열 데이터의 앞뒤에 자동적으로 2중 인용부호가 삽입된다.

③ 수치 데이터를 컴마로 구분하여 출력해도 공백을 생략하고 출력한다.

PRINT# 문으로 디스크에 기록하여, INPUT #문으로 재입력할 때 하나의 PRINT #문으로 하나의 데이터만 출력하든지, 수치 데이터만 출

력할 때는 문제가 없지만, 문자열 출력 항목이 두개가 이어지거나 하면 구분이 된다.

이것을 피하기 위하여 지금까지 PRINT # 문의 출력 항목 구분마다 컴마나 2중 인용부호를 문자 정수형으로 삽입해 주어야 하는데, 이 시스템에서는 WRITE # 문을 사용하면 가능하게 된다.

〈예〉 데이터의 기록, 재생

데이터 a_1, a_2,…, a_n을 키보드에서 입력하여 디스크에 기록하는 프로그램과 기록된 데이터를 재생하는 프로그램을 작성하는 예제

〈해〉 데이터를 키보드에서 입력하는 데는 상당히 수고스러우므로 한번 입력한 데이터는 가능하면 디스크에 기록하여 보존하는 것이 좋다. 여기에 보여주는 예제는 이와 같은 프로그램의 간단한 본보기이며, 시간이 경과한 다음에 데이터를 식별하기 위하여 표제(title)를 붙여 두기로 한다. 데이터의 개수 N도 기록하여 두는 것이 편리하며, 화일명은 키보드에서 입력하도록 하였으며, 이렇게 하지 않으면 같은 이름의 화일이 많이 생긴다.

〈프로그램〉 데이터의 기록과 재생

```
 10 REM --- INPUT ---
 20 "INPUT input or output ? (I/O)" ; IO$
 30 IF IO$="I" GOTO 200
 40 INPUT "titlie" ; TILE$
 50 INPUT "데이터 수는"; N
 60 DIM A(N)
 70 FOR I=1 TO N
 80   INPUT A(I)
 90 NEXT I
 100 REM --- SAVE ---
 110 INPUT "화일명은"; FILE NAME$
 120 OPEN FILE NAME$ FOR OUTPUT
AS #1
 130 WRITE #1, N
 140 WRITE #1 TO N
 160   WRITE #1, A(I)
 170 NEXT I
 180 CLOSE #1
 190 END
 200 REM --- LOAD ---
 210 INPUT"화일명은"; FILE NAME$
 220 OPEN FILE NAME$ FOR INPUT AS
#1
 230 INPUT #1, TITLE$
 240 PRINT TITLE$
 250 INPUT #1, N
 260 DIM A(N)
 270 FOR I=1 TO N
 280   INPUT #1, A(I)
 290   PRINT A(I)
 300 NEXT I
 310 CLOSE #1
 320 END

run
input or output ? (I/O) ? 0
title ? TEST
데이터 수는 ? 6
? 10
? 20
? 30
? 100
? 200
? 150
화일명은 ? A B C
OK
run
input or output ? (I/O) ? I
화일명은 ? A B C
TEST
 10
 20
 30
 100
 200
 150
OK
```

2. 직접처리 방법

직접처리(random access)라는 것은 데이터를 임의의 순서로 읽고 기록하는 것이다. 이에 대해서 항상 일정한 순서로 읽고 기록하는 것을 순서적 처리라고 한다. 직접처리를 하기 위해서는 기억장소에 번지를 붙여두고 지정된 번지의 데이터를 꺼내서 지정된 번지에 기록할 수 있도록 하면 된다.

그러나, 디스크일 때 수백 키로 바이트(KB)의 기억장소에 1 바이트마다 번지를 붙이는 것은 어려운 일이며, 1 바이트 단위로 입출력을 하는 것은 기다리는 시간 관계로 비경제적이므로 더욱 큰 처리단위를 만들어서 그 단위마다 번지를 붙여서 정리하여 입출력하고 있다. 이 처리단위를 레코드(record)라고 한다.이 시스템의 레코드 크기(1 record size)는 OPEN 문에 의하여 자유롭게 설정할 수 있다. 디스크의 입출력에는 언제나 출력시에는 각각의 데이터를 큰 단위로 정리하고 입력시에는 큰 단위로 읽은 데이터를 각각의 데이터로 분류하는 작업이 필요하게 된다. 이를 위한 작업장소를 buffer라고 한다. 순서적 처리 방식일 때는 이들의 작업은 모두 시스템에서 처리하여 준다.

PRINT # 문으로 출력된 데이터는 먼저 버퍼(buffer)에 들어가서 1기록 단위분이 정리되면 자동적으로 디스크에 전송된다. 입력시에도 이와 같이 자동적으로 처리되므로 사용자는 버퍼를 의식할 필요가 없으나, 직접처리 일 때는 자동적으로 처리하여 주지 않으므로 사용자가 스스로 정리와 분류 프로그램을 작성해 주어야 한다.

이 시스템에서는 다음과 같이 사용한다.

① OPEN 문에 의하여 화일을 연다.

② FIELD 문에 의하여 레코드를 필드(Field)로 나눈다.

③ 수치 데이터를 문자열로 변환한다.

④ 그 결과를 버퍼에 수록한다.

⑤ PUT 문에 의하여 디스크에 기록한다.

⑥ GET 문에 의하여 디스크에서 1 레코드 읽는다.

⑦ 필드별로 문자열 데이터로 꺼낸다.

⑧ 필요한 경우에 그것을 수치 데이터로 변환한다.

⑨ CLOSE 문에 의하여 화일을 닫는다.

2-1. 직접처리 화일 OPEN

일반형은 다음과 같다.

OPEN "화일명" AS #번호 LEN=레코드 길이

〈예 OPEN "B : NOTE"AS #1 LEN=256

이 방법외에

OPEN "R," #번호,"화일명," 레코드 길이

라고 할 수 있으며, 이것을 다음과 같이 써도 같다.

OPEN "R, #1, "B : NOTE," 256

2-2. FIELD 문

FIELD 문의 일반형은 다음과 같다.

FIELD # 화일번호, 길이 AS 문자열 변수,…, 길이

AS 문자열 변수

〈예 FIELD #1, 10 AS A$, 2 AS K$

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