4-1강

변수의 정의

- INPUT문 (자료의 입력 및 변수의 지정)
  - INPUT 문은 DATA문을 기술한 다음입력했거나 또는 앞으로 입력할 자료를 해당 변수에 할당하여 읽고 기억시키는 절차

  - 이 문장에서 변수명을 지정

  - 데이터의 입력방식은

    1. 자유형식 입력 방식
    2. 고정 입력 방식
    3. 포인터 입력 방식
    4. 혼합 입력 방식

SAS data step

```
(1) data one;
(2) input x y;
(3) z=x+y;
(4) cards;
    1 2
    3 4
```

- 정확히 이해하라

1. data step의 시작

   (변수 x, y, z에 대하여 buffer(임시기억장소)를 만들고 그 기억장소에 결측값을 저장 한다.) (1)

2. input 문을 수행한다. (자료 1, 2를 읽고 변수 x와 y의 buffer에 저장 한다.) (2)

3. 덧셈을 수행한다. (변수 x와 y의 buffer 내용을 더하여 변수 z의 buffer에 저장한다.) (3)

4. data step의 문장이 더 이상 없으므로 buffer의 내용을 data set에 저장하고 data step의 시작 으로 간다. (4)

5. buffer의 모든 내용을 결측값으로 만든다. (1)

6. input 문을 수행한다. (자료 3, 4를 읽고 변수 x 와 y의 buffer에 저장한다.) (2)

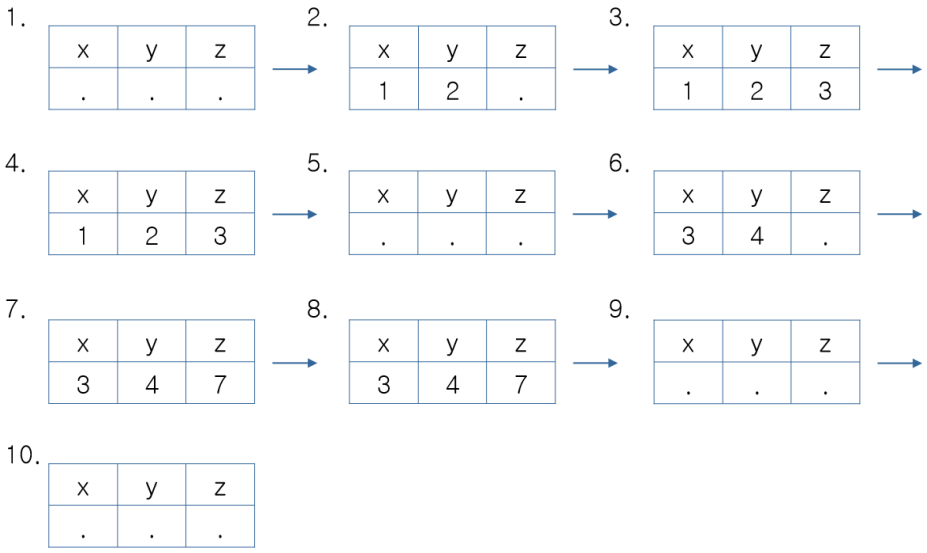
7. 덧셈을 수행한다. (변수 x와 y의 buffer 내용을 더하여 변수 z의 buffer에 저장한다.) (3)

8. data step의 문장이 더 이상 없으므로 buffer의 내용을 data set에 저장하고 data step의 시작 으로 간다. (4)

9. buffer의 모든 내용을 결측값으로 만든다. (1)

10. input 문을 수행하는 중 더 이상의 읽을 자료가 없으므로 data step을 끝낸다.

결측값은 dot(.)을 의미한다.



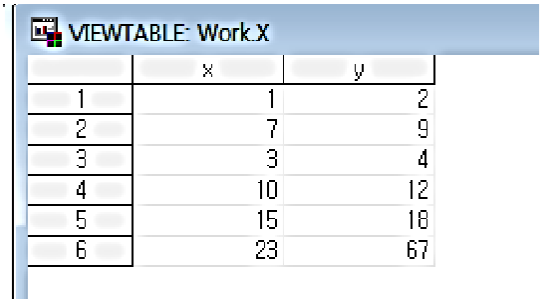
텍스트 자료에서 자유형식 (free format) INPUT문

```
INPUT variables $ ;
```

- 가장 단순한 INPUT 형식으로 변수를 나열한 순서대로 읽음
- 변수(또는 자료)의 구분은 자료에 있는 공백으로서 구분
- 만약 결측값이 있다면 자료를 입력할 때 소수점(.)으로 표시해 주어야만 결측치로 인식

- 문자 변수(변수에 할당되는 실제 값을 모두 문자로 인식)는 $ 표시를 변수 뒤에 해주어야 문자 변수로 정확히 읽어 들일 수 있음
- 숫자변수는 아무런 표시가 없어도 됨

```
 DATA x;
INPUT x y @@;
CARDS;
1 2 7 9 3 4 10 12
15 18 23 67
;
```



	x	y
1	1	2
2	7	9
3	3	4
4	10	12
5	15	18
6	23	67

@@은 끝까지 다 읽어주는 것이다.

- INPUT 문의 마지막에 있는 @@ 표시는 한 데이터 라인에서 여러 개의 관측치를 중복해서 읽을 때 주로 사용
- 모든 데이터 값이 읽혀질 때까지 현 데이터 라인을 계속 읽음 (@는 “ at- sign” , ‘앳’라고 읽는다)
- 만일 그 라인의 자료를 전부 읽어 들인 경우는 다음 라인의 처음부터 다시 계속해서 읽어 들이며 더 이상의 자료가 존재하지 않으면 자료의 입력을 끝마침

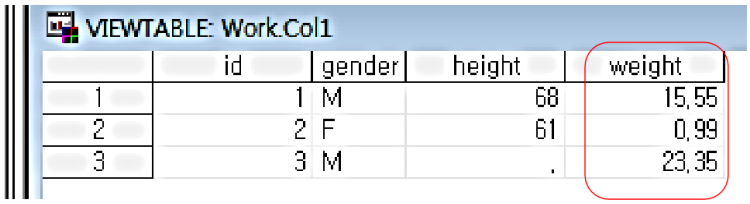
고정 입력 방식 (fixed format) INPUT문

```
INPUT variables $ 시작열(column) - 끝열 [.decimal] …..
```

- 이 형식은 자료가 각각 몇 열 (column) 에서 몇 열 사이에 있는지 알기만 하면 공백에 상관없이 그 열에 해당하는 관측값을 읽을 수 있음
- 또 자료를 입력 할 때 번거롭게 소수점을 입력해 줄 필요 없이 소수점 이하 자릿수를 소수점(.) 다음에 표시하면 자연스럽게 읽어 들일 수 있음
- 예를 들어 x 1-5 .2 는 1열에서 5열 사이의 숫자를 x 로 받아들이는데 여기서 소수점 이하 두 자리의 형태로 읽어 들이라는 뜻

```
DATA col1;
INPUT id 1-3 gender $ 4 height 5-6 weight 7-11 .2;
CARDS;
```

```
001M681555
2  F61   99
  3M   2335
;
RUN;
```



	id	gender	height	weight
1	1	M	68	15,55
2	2	F	61	0,99
3	3	M	.	23,35

포인터 (pointer) INPUT문

```
INPUT variable $ @열 variable [n.] ..... [#n] ... ;
```

- 포인터 INPUT 형식은 현재 읽고 있는 위치에서 @ (at- sign) 다음에 오는 열로 이동해서 읽을 수 있는 명령
- # (샤프) 다음에 오는 #n은 n번째 라인의 첫 열로 이동하라는 명령

예 1) INPUT @4 number 3.;

현재 라인의 4열로 포인트를 이동하여 그로부터 세 칸을

즉, 4 - 6 열 안에 있는 자료를 number 라는 변수로 읽음

여기서 3. 과 3은 분명한 차이가 있음

만약 3. 대신 소수점이 없는 3을 쓰면 3열로 인식하게 되어 에러 발생

@ 다음에 아무런 숫자도 없이 그냥 @ 하나만 사용하는 것은,

현재의 포인터에서 읽기를 잠시 중지하고 다음 명령을 기다리라는 표시

예 2) INPUT aaa 9-10 #2 bbb 3-4;

..... 현재 라인에서 aaa라는 변수를 9-10 열에서 읽고,

다음 라인의 3-4 열에서 bbb 를 읽음

예 3) INPUT @10 x1 / x2 10-11 + 2 x3;

열 번째 열로 포인터를 이동하여 x1을 읽고 다음 라인의 10-11 열에서 x2 를 읽음.

그로부터 두 칸 즉 13 열로 이동하여 x3을 읽음

(여기서 / 는 현재 라인의 다음 라인을 말함. 여기서는 #2와 동일한 효과.

+n는 n칸만큼 오른쪽으로 포인터를 이동하라는 명령어)

예 4) INPUT @10 x1 / x2 10-11 + (-2) x3 3.1;

위의 예 3)과 비슷하지만 x2 를 10-11 열에서 읽고, 왼쪽으로 두 칸 이동하여,

즉 9 열에서 x3을 3.1 로 읽으라는 명령

즉 한 번 읽은 열을 되돌아가서 다른 형식으로 다시 읽을 수 있다는 것

이 예는 INPUT @10 x1 / x2 10-11 @9 x3 3.1; 와 동일

예 5) 모두 동일하게 데이터를 읽는 프로그램

X1-X3 변수를 2자리로 읽고 다음 줄에 y1-y4 변수를 6 자리 정수로 읽음

```
DATA one
INPUT (x1-x3)(3*2.) #2 (y1-y4)(4*6.);
```

```
DATA one
INPUT (x1-x3) (2.) #2 (y1-y4) (6.);
```

```
DATA one ;
INPUT x1 1-2 x2 3-4 x3 5-6/
y1 1-6 y2 7-12 y3 13-18 y4 19-24;
```

혼합 INPUT 형식

- 위에 정의한 여러 가지 INPUT 형식을 혼용하여 자료를 읽어 들일 수 있음

예 1) INPUT name $ 10. (a1-a20) (2.);

..... name 이라는 문자변수를 처음부터 10열까지 읽고 난 뒤

a1 부터 a20 까지는 두 열씩 읽음

위의 (a1-a20) (2.)는 (a1-a20) (20*2.)과 같은 의미

만약 괄호를 생략하고 a1-a20 (2.) 라고 쓰면 에러가 발생

예 2)

```
DATA point;
INPUT @1 id 3. @4 gender $ 1. @9 age 2. @11 weight 2. @16 v_date 4.;
CARDS;
001M000016817501022
002M222245762990331
RUN;
```

VIEWTABLE: Work.Point

	id	gender	age	weight	v_date
1	1	M	16	81	1022
2	2	M	45	76	331

..... @1 은 1열로 이동하라는 명령이고 3. 은 1열부터 세 칸을 읽으라는 명령

주어진 자료에서 포인트를 이용하여 필요 없는 자료를 읽지 않고 선택적 으로 원하는 자료만 읽을 수 있음

예 3)

```
DATA pointer;
INPUT @1 id 3. @8 weight 2. @4 gender $ 1. @11 category 1.;
CARDS;
001M1681750
002M4576299
RUN;
```

VIEWTABLE: Work.Pointer

	id	weight	gender	category
1	1	17	M	0
2	2	62	M	9

포인터를 이용하여 필요에 의해서 변수의 순서 바꾸어 읽음 한번 읽은 자료를 다시 읽을 수도 있음

예 4)

```
DATA question;
INPUT year 1-10 @;
IF year =1996 THEN INPUT @11 (x1-x10)( 1.);
ELSE IF year = 1990 THEN INPUT @15 (x6-x10)( 1.);
CARDS;
```

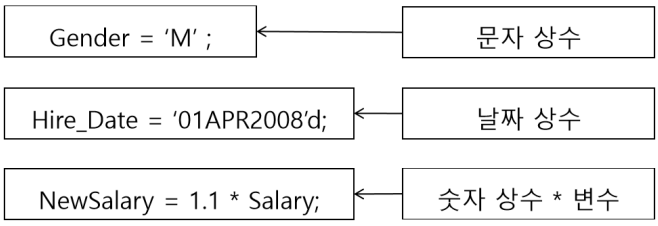
- @ 다음에 아무런 숫자도 없이 그냥 @ 하나만 사용하는 것은, 현재의 포인터에서 읽기를 잠시 중지하고 다음 명령을 기다리라는 표시

변수 생성

할당 문장

```
Variable = 표현식;
```

- 새로운 변수를 추가하거나 기존 변수의 값을 갱신하기 위해 사용
- 등호 왼쪽은 새로운 또는 기존 변수 이름, 오른쪽은 왼쪽변수에 저장할 값을 계산하기 위한 표현식 지정
- 표현식을 작성할 때는 연산자와 피연산자를 사용



산술할당문

- SAS 식은 사용자가 가장 알기 쉬운 방법으로 기술하게 되어 있음
- 즉 익숙하게 보아온 연산자와 명령의 결합으로 이루어져 있으므로 함수의 사용, 괄호 등으로 원하는 값을 유도해 낼 수 있음
- 예를 들어 어떤 변수 y에 일률적으로 10을 더한 값을 새로운 변수로 하고 싶을 경우

⇒ 새로운 변수 y1을 가정하면 y1 = y + 10; 이라 써주면 y1은 y에 10을 더한 값이 할당

- 등호 오른쪽의 산술식이 왼쪽의 변수에 할당되는 형태로서, 등호 왼쪽에는 산술식이 나와서는 안됨

예 1) x1= log(x); y = b/a; y= a*(c+3);

새로운 변수를 등호 오른쪽에 기술한 산술식으로 할당, 연산자 우선 순위에 따라서 연산

예 2) name = 'kim';

name이라는 변수를 새로 만들어 kim 이라는 문자 상수 할당

예 3) DATA kim; INPUT a b; c= a+b;

원래 자료에는 c라는 변수 값을 입력하지 않았지만 a에 b 를 더한 값들로 c를 만들어 주어 데이터 셋 kim 에 포함