통계계산소프트웨어

데이터 단계에서 사용되는 명령문

2018. 9.

차 례

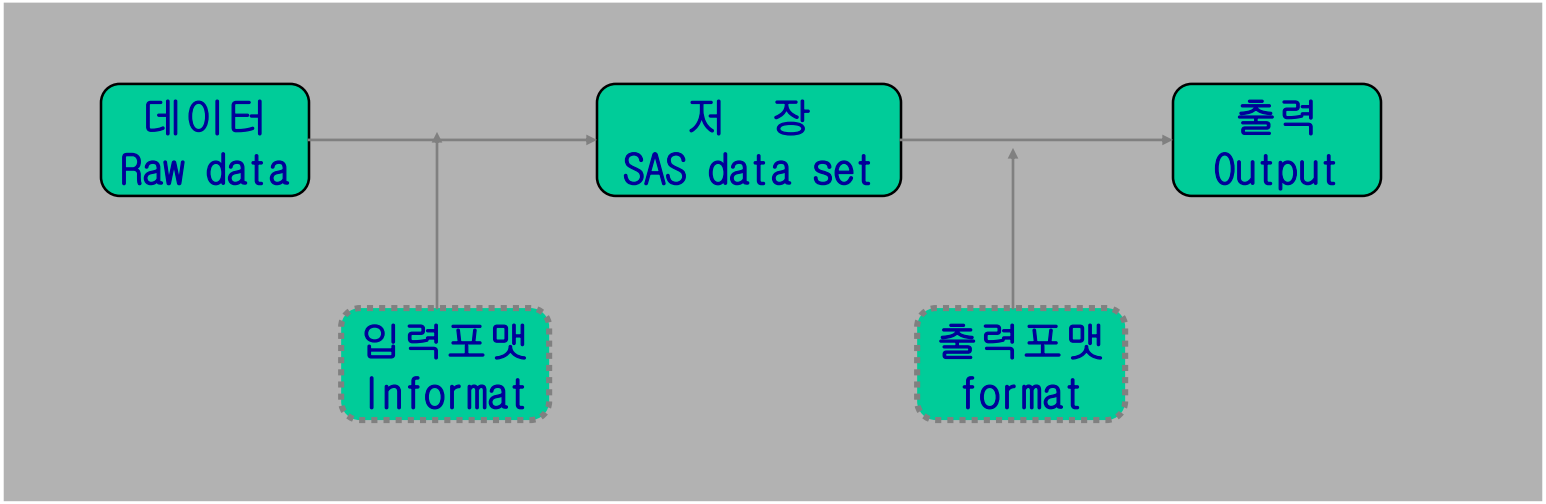
Review

INPUT 명령문

변수 지정 데이터 읽기 – Input 명령문

	List INPUT	Column INPUT
구 분	하나 이상의 공백	Column번호로 구분
자료값	8자리 까지의 값만 가능 (LENGTH 명령문 이용하여 최대 200자 까지 저장가능)	최대 200자리의 값 가능
결측값	꼭 '.'으로 표시하여야 한다	공백으로 처리할 수 있다.
문자변수	문자 변수내에 공백문자를 포함할 수 없다.	문자 변수내에 공백문자를 포함할 수 있다.
기 타		읽은 열의 위치가 지정되므로 어떤 순서로 읽어도 무방하며, 같은 열을 반복하여 읽을 수 있다.

입력포맷과 출력포맷의 처리절차



자료	입력포맷	저장	출력포맷	출력
abcdefghij	$10.	abcdefghij	$5.	abcde
1.23456789	10.	1.23456789	4.2	1.23
1,100,100	COMMA9.	1100100	DOLLAR10.	$1,100,100
01JAN1998	DATE9.	13880	YYMMDD8.	98-01-01

데이터입력 구분자

구분자 지정
ㅇ 구분자 - DELIMITER=문자열 - DLM=문자열 - DLMSTR=문자열 - 여러 개의 문자를 지정 - 연속된 구분자는 1개로 간주 - 기본 구분자는 공백 ㅇ 정밀한 구분 : DSD - 연속된 구분자는 결측값 포함 - 문자열 “ “로 표시 - 기본 구분자는 ‘,’ ㅇ Tab구분자(엑셀) : EXPANDTABS

사용 위치
INFILE fileref DLM=‘ab ,‘ dsd ; FILE fileref DLM=‘ab ,‘ dsd ;

☞ *확인 팁*
- *ㅇdelimiter는 공백으로 변환한다*
- *ㅇ dsd가 있는 경우 공백사이에 . 을 추가*
  공백이 둘이면 공백.공백 으로
  공백이 셋이면 공백.공백.공백

예제: 구분자

다음의 출력결과는 ?

```
DATA nums1;
     INFILE CARDS DLM='ab';
     INPUT x y z;
DATALINES;
1aa2ab3
4b5bab6
7a8b9
;RUN;
PROC PRINT;RUN;

DATA nums2;
     INFILE CARDS DSD DELIMITER='ab';
     INPUT   x y z ;
CARDS;
1aa2ab3
4b5bab6
7a8b9
;RUN;
PROC PRINT;RUN;
```

구분자 : a b

```
1  2  3
4 5   6
7 8 9
```

```
1 2 3
4 5 6
7 8 9
```

Obs	x	y	z
1	1	2	3
2	4	5	6
3	7	8	9

구분자 : a b
정밀한 구분

```
1  2  3
4 5   6
7 8 9
```

```
1 . 2 . 3
4 5 .  . 6
7 8 9
```

Obs	x	y	z
1	1	.	2
2	4	5	.
3	7	8	9

부족한 자료

ㅇ MISSOVER : 데이터가 부족한 변수는 결측값으로 할당해서 다음 레코드를 읽는 것을 방지

ㅇ TRUNCOVER : 데이터값이 부족한 부분에 공백을 삽입하라

ㅇ FLOWOVER : 데이터값이 부족하면 다음레코드를 읽어서 INPUT Buffer에 넣어라

ㅇ STOPOVER : 데이터값이 부족하여 결측값이 생기면 데이터 생성을 중단한다.

※ 네가지 중에서 한가지만 지정할 수 있으며, default는 FLOWOVER

예 5.12 MISSOVER 옵션의 사용

MISSOVER :자룟값을 읽지 못하는 변수에 대해서 결측값으로 처리

```
DATA   miss;
   INFILE   CARDS   MISSOVER;
   INPUT a   b   c ;
CARDS;
1    2    3
4    5
6    7    8
9    0    1    2
3    4    5
;
RUN;
```

[프로그램 내부에 데이터를 입력하는 경우 일반적으로 INFILE 명령문을 사용할 필요가 없지만 INFILE 명령문에 있는 옵션을 사용하기 위해서 INFILE CARDS 명령문을 사용할 수도 있음]

	a	b	c
1	1	2	3
2	4	5	.
3	6	7	8
4	9	0	1
5	3	4	5

예 5.13 DLM 또는 DSD 옵션의 사용

```
DATA   score;
   INFILE   CARDS   DLM=' , ';     (또는 INFILE CARDS DSD)
   INPUT t1 t2 t3;
CARDS;
91,87,95
97,,92          /* 비교 97, ,92 */
1,1,1
;
RUN;
```

DSD 옵션
1. 콤마(,)를 구분자로 사용한다. 콤마 이외 구분자는 DLM옵션과 같이 사용
2. 연속된 구분자들 사이는 결측값으로 처리
3. 인용부호안에 있는 문자는 구분자로 인식안함. 인용부호는 자룃값으로 받아들이지 않음

VIEWTABLE: Work.Score

	t1	t2	t3
1	91	87	95
2	97	.	92
3	1	1	1

	t1	t2	t3
1	91	87	95
2	97	92	1

1. 할당문과 선언문

- 할당문

  ✓사용자는 다음과 같이 새로운 변수 혹은 기존 변수를 생성하거나 값을 바꿀 수 있다.

Variable = Expression

Example

major = '통계학'	문자형 변수의 값을 할당하는 경우
mean = (test1 + test2)	기존 변수와의 연산을 통해 값을 할당하는 경우
x = log(y)	함수를 이용하여 값을 할당하는 경우

- 선언문

  ✓지정된 변수에 대해 특정한 기능을 표시한다.

  ✓LENGTH, FORMAT, LABEL, DROP, KEEP, RENAME 등이 있음.

예6.1 : 다양한 할당문의 사용

```
DATA exam;
     INPUT id $ name $ mid final;
     x=30;                          /* Numeric Constant       */
     major='통계학';                /* Character Constant     */
     major_id='STA'||id;            /* Concatenation          */
     mean=(mid+final)/2;            /* Arithmetic Expression  */
     y=mid**3;                      /* Exponentiation         */
     LABEL name='이름'
           mid='중간성적'
           final='기말성적';
CARDS;
001 김철수 10 40
002 이영희 15 10
001 강민호 50 15
001 박지수 20  .
;
RUN;
```

[연결연산자(||)는 여러 개의 문자열과 문자변수들을 연결하는 연산자로 공백문자는 제거되지 않는다]

VIEWTABLE: Work.Exam

	id	이름	중간성적	기말성적	x	major	major_id	mean	y
1	001	김철수	10	40	30	통계학	STA001	25	1000
2	002	이영희	15	10	30	통계학	STA002	12.5	3375
3	001	강민호	50	15	30	통계학	STA001	32.5	125000
4	001	박지수	20	.	30	통계학	STA001	.	8000

2. 함수의 이용: 숫자함수

✓여러 가지 기능을 갖는 함수를 이용하여 복잡한 형태의 연산이나 필요로 하는 변수 변환을 손쉽게 수행할 수 있다.

Variable = function_name([[arg1] [,arg2, …])

숫자 함수

y = sum(OF x1-x10)	변수 x1에서 x10까지의 합을 구하여 변수 y에 할당한다..
y = mean(x1, x2, x3)	변수 x1, x2, x3의 평균을 계산하여 변수 y에 할당한다.

여러 가지 함수의 사용법과 소개는 **SAS**의 **HELP** 참조.

함수의 이용: 문자함수와 날짜함수

형 식	기 능
compress(*arg, chars*)	arg의 내용 중 chars로 주어진 문자를 제거
left(*arg*)	주어진 arg의 앞 공백을 모두 제거
length(*arg*)	주어진 arg의 문자열 길이를 계산
substr(*arg, p, n*)	arg의 p번째 문자로부터 n개의 문자열을 선택
translate(*arg, to, from*)	주어진 arg의 문자열에서 from부분을 to로 변환
trim(*arg*)	주어진 arg의 오른쪽 공백문자를 모두 제거
DAY(*date*)	Date의 값을 구한다.(1~31)
MDY(*month, day, year*)	년, 월, 일에 해당하는 SAS 날짜값을 계산한다.
MONTH(*date*)	Date의 월을 구한다.(1~12)
YEAR(*date*)	Date의 년도를 구한다.
WEEKDAY(*date*)	Date의 요일값을 구한다.(1~7)
TODAY()	현재 날짜에 해당하는 SAS날짜값을 구한다.
INTNX(*'interval',from,n*)	from부터 n interval이후 날짜를 계산한다.

예6.2 : 다양한 함수의 이용

```
DATA exam1;
     INPUT name $ 1-8 mid final enterm enterd;
     mid=INT(mid);                    /* Numeric Function   */
     max_mf=MAX(mid,final,30);
     name=COMPRESS(name);           /* Character Function */
     f_name=SUBSTR(name,1,2);
     ent_day=MDY(enterm,enterd,1999); /* Date Function */
     ent_week=WEEKDAY(ent_day);
     FORMAT ent_day yymmdd6.;
CARDS;
김 철 수 10.1 40 11 22
이  영희 15.7 10 11 29
강민호   50.3 15 12 05
박지  수 20.8  . 12 15
;
RUN;
```

VIEWTABLE: Work.Exam1

	name	mid	final	enterm	enterd	max_mf	f_name	ent_day	ent_week
1	김철수	10	40	11	22	40	김	991122	2
2	이영희	15	10	11	29	30	이	991129	2
3	강민호	50	15	12	5	50	강	991205	1
4	박지수	20	.	12	15	30	박	991215	4

예6.3 : MEAN과 SUM 함수의 사용

```
DATA score;
  INPUT name $ (x1-x3) (1.) y1-y3;
  total = SUM(OF x1-x3 y1-y3);
  average  = MEAN(OF x1-x3 y1-y3);
  logx = LOG(x1);
  sqrtx = SQRT(x1);
  intmean = INT(average);
CARDS;
김철수 551 2 1 3
최민지 .31 4 5 1
이영희 153 2 . 2
오인수 412 4 . .
;
RUN;
PROC PRINT DATA=score;
RUN;
```

SAS 시스템

OBS	name	x1	x2	x3	y1	y2	y3	total	average	logx	sqrtx	intmean
1	김철수	5	5	1	2	1	3	17	2.83333	1.60944	2.23607	2
2	최민지	.	3	1	4	5	1	14	2.80000	.	.	2
3	이영희	1	5	3	2	.	2	13	2.60000	0.00000	1.00000	2
4	오인수	4	1	2	4	.	.	11	2.75000	1.38629	2.00000	2

3. 조건문: IF-THEN-ELSE 명령문

✓어떤 작업 수행을 모든 관찰치에 대하여 수행하는 것이 아니라 특정 조건에 부합하는 자료만을 처리하기 위한 명령문

IF *condition* THEN *action*; [ELSE [*action*;]]

조건 논리식 — 명령문

✓비교연산자

기호	약어	기능	기호	약어	기능
=	EQ	같다	>=	GE	크거나 같다
^= 또는 ~=	NE	같지 않다	<=	LE	작거나 같다
>	GT	크다	&	AND	그리고
<	LT	작다	| 또는 !	OR	또는

✓IN 옵션 - IF문을 사용하는 데 있어 여러 개의 자료값의 범위를 지정해야 하는 경우에 유용하게 사용할 수 있는 option

IF *variable* IN *(arg1 [,] agr2 [, arg3 , …])* THEN *action;*

조건문 : if ~ else

- if (조건문) then statement;

  조건문이 참이면 statement를 수행하게 되고 그렇지 않으면 수행하지 않는다.

예)

```
data one;
    input x y;
    if x<y then x=x-1;
        cards;
12 13
14 15
11  8
;
```

결과:

x	y
11	13
13	15
11	8

```
data one;
    input x y;
    if x>13 then z=x-1;
    cards;
12 13
14 15
11  8
;
```

결과:

x	y	z
12	13	.
14	15	13
11	8	.

조건문 : if ~ else

❑ 주의 :

if문에서 조건이 참일 때 if문에 있는 문장 하나만을 수행한다. 따라서 그 다음에 오는 문장들은 if문에 영향을 받지 않는다. if문에서 조건이 참일 때 두 개 이상의 문장을 수행하고 조건문이 거짓이면 두 문장 다 수행하지 않게 하려면 do;... end;를 사용하면 된다.
(하나의 조건식에 대해 여러 개의 할당문을 동시에 지정)

예)

```
data one;
    input x y;
    if x<13 then x=x-1;
                y=y-1;
    cards;
12 13
14 15
11  8
;
```

결과:

x	y
11	12
14	14
10	7

조건문 : if ~ else

예)

```
data one;
    input x y;
    if x<13 then do;
       x=x-1;
       y=y-1;
    end;
    cards;
12 13
14 15
11  8
;
```

결과:

x	y
11	12
14	15
10	7

조건문 : if ~ else

- else

  else문은 if문 바로 다음에 위치하여야 한다.

  if (조건문) then statement1;
  else statement2;

  만약 조건문이 참이면 statement1을 수행하고 거짓이면 statement2를 수행한다.

  예)

```
data one;
    input x y;
    if x < y then z=x-1;
    else z=y-1;
    cards;
12 13
14 15
11  8
;
```

결과:

x	y	z
12	13	11
14	15	13
11	8	7

주의) if문과 else문에서 오로지 한 문장만이 영향을 받는다.

조건문 : if ~ else

- if else의 연결

if c_1 then s_1 ;
else if c_2 then s_2 ;
else if c_3 then s_3 ;
…
else if c_{k-1} then s_{k-1} ;
else s_k ;
(statement)

- 만약 c_1이 참이면 s_1을 수행하고 (statement)로 가서 수행한다. 만약 c_1이 거짓이면 else에 해당되는 if문을 수행한다. 즉, c_2가 참이면 s_2를 수행한 후 (statement)로 가서 수행하게 되고 거짓이면 그 다음의 else문의 if문에서 c_3 가 참이면 s_3를 수행한 후 (statement)로 가서 수행하고 거짓이면 else문을 수행한다.…
  c_{k-1}이 참이면 s_{k-1}을 수행한 후 (statement)로 가서 수행하게 되고 거짓이면 그 다음의 else문의 s_k를 수행하고 (statement)를 수행한다.

조건문 : if ~ else

- Note : if 문은 else 문까지 하나의 block으로 생각할 수 있다.

예)

```
data grade;
    input id score;
    if score >= 90 then grade='A';
    else if score >= 80 then grade='B';
    else if score >= 70 then grade='C';
    else if score >= 60 then grade='D';
    else grade='F';
    cards;
123 89
456 92
376 73
129 55
;
```

결과:

id	score	grade
123	89	B
456	92	A
376	73	C
129	55	F

조건문 : if ~ else

주의)

```
data grade;
    input id score;
    if score >= 90 then grade='A';
    if score >= 80 then grade='B';
    if score >= 70 then grade='C';
    if score >= 60 then grade='D';
    else grade='F';
    cards;
123 89
456 92
376 73
129 55
;
```

결과:

id	score	grade
123	89	D
456	92	D
376	73	D
129	55	F

조건문 : if ~ else

수정)

```
data grade;
    input id score;
    if score >= 90 then grade='A';
    if score < 90 & score >= 80 then grade='B';
    if score < 80 & score >= 70 then grade='C';
    if score < 70 & score >= 60 then grade='D';
    if score < 60 then grade='F';
    cards;
123 89
456 92
376 73
129 55
;
```

결과:

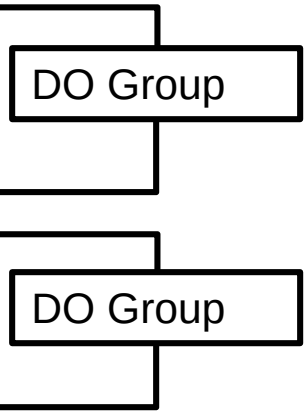
id	score	grade
123	89	B
456	92	A
376	73	C
129	55	F

조건문

- IF-THEN / ELSE 문장

```
IF       조건표현식 THEN 실행문;
ELSE IF 조건표현식 THEN 실행문;
ELSE     실행문;
```

```
IF 조건표현식 THEN DO;
    실행문장(들)
END;
ELSE IF 조건표현식 THEN DO;
    실행문장(들)
END;
ELSE;
```



예6.4 : IF-THEN 명령문의 사용

```
DATA exam;
   INPUT id $ name $ mid final;
CARDS;
001 김철수 10 40
002 이영희 15 10
001 강민호 50 15
001 박지수 20  .
;
RUN;
DATA exam2; SET exam;
 IF final=. THEN final=10;
 IF (mid+final)>=50 THEN score1= ' P'  ;
 IF mid>=30 or final>=30 THEN score2= 'P';
 IF score1=' ' THEN
   DO;
     score1='F' ;
     score2='F' ;
   END;
RUN;
```

	id	name	mid	final	score1	score2
1	001	김철수	10	40	P	P
2	002	이영희	15	10	F	F
3	001	강민호	50	15	P	P
4	001	박지수	20	10	F	F

조건문_실습

- 예제
  - √ 데이터 : 20명의 통계학과 수강생들에 대한 기초조사 결과
  - √ 연령(세), 성별(1 남자, 2 여자), 키 (cm), 체중(kg),
  - √ 즐기는 음식(1 육류, 2 생선류, 3 채소류)을 조사한 결과

  [Data] d:₩data₩sample1.txt

- SAS 데이터 셋을 생성 후 아래의 3개 변수를 추가 생성

연령 조건	새로운 변수
	Age_range(문자 10자리)
<30	20대
<35	30대초반
<40	30대후반

조건문_실습

- 실습 (SAS code)

```
DATA sample1;
INPUT index age gender height weight food;
CARDS;
1 30 1 183 82 1
2 28 2 160 62 3
  (중간 생략)
18 26 1 166 69 3
19 26 1 169 66 2
20 28 2 159 60 2
;
PROC PRINT;
RUN;

DATA p3_ex;
SET sample1;
LENGTH age_range $ 10;
IF age<30 then age_range= '20대' ;
else if age<35 then age_range='30대초반';
else age_range='30대후반';
PROC PRINT;
RUN;
```

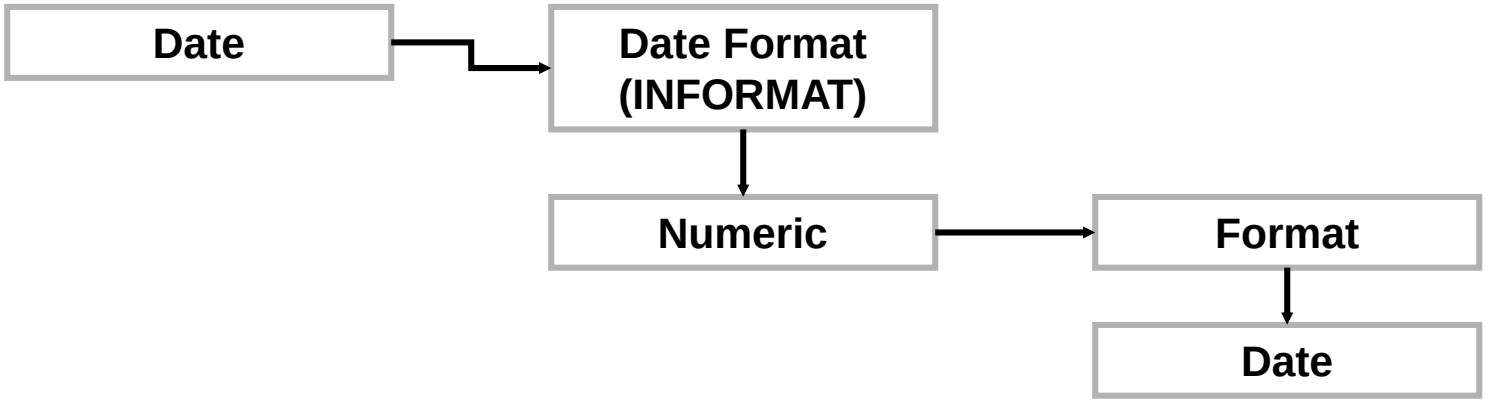
조건문_실습

- 실습 (SAS output)

OBS	index	age	gender	height	weight	food	age_range
1	1	30	1	183	82	1	30대 초반
2	2	28	2	160	62	3	20대
3	3	27	1	178	77	2	20대
4	4	23	1	172	70	2	20대
5	5	25	1	168	72	3	20대
6	6	27	1	179	77	1	20대
7	7	26	1	169	71	1	20대
8	8	29	1	171	75	3	20대
9	9	34	2	158	60	2	30대 초반
10	10	31	1	183	77	3	30대 초반
11	11	26	2	162	59	1	20대
12	12	26	1	173	70	2	20대
13	13	35	1	173	68	3	30대 후반
14	14	24	1	176	66	3	20대
15	15	29	2	170	70	2	20대
16	16	33	1	177	72	2	30대 초반
17	17	38	2	159	55	1	30대 후반
18	18	26	1	166	69	3	20대
19	19	26	1	169	66	2	20대
20	20	28	2	159	60	2	20대

4. 날짜변수

✓SAS내의 날짜 처리 방법



★ 날짜변수는 입력포맷(informat)으로 변수를 읽어드림
★ 날짜상수 : 'ddmmmyy'D 형태로 지정
★ 숫자화되어 저장되어 있는 날짜변수를 사용자가 보기 편한 형태의 날짜형식으로 출력하거나, 보기 위해서는 FORMAT 명령문에 의해 날짜변수의 출력포맷을 지정해 주어야 한다

날짜변수 입력포맷

입력포맷	자료표현	사용예	w
DATEw.	일/월이름/연도	13FEB98	7
DDMMYYw.	일/월/연도	13-02-98	8
MMDDYYw.	월/일/연도	02/13/98	8
YYMMDDw.	연도/월/이름	98-02-13	8
YYMMDDw.	연도/월/이름	1998-02-13	10
MONYYw.	월이름/연도	FEB98	5
JULIANw.	연도/날짜수	1998044	7
YYQw.	연도/분기	98/2	4

날짜변수 출력포맷(PUT문)

```
DATA d;
birth=8966;
put birth date7.;
put birth date9.;
put birth day2.;
put birth day7.;
put birth mmddyy8.;
put birth mmddyy6.;
put birth weekdate15.;
put birth weekdate29.;
run;
```

```
15   DATA d;
16   birth=8966;
17   put birth date7.;
18   put birth date9.;
19   put birth day2.;
20   put birth day7.;
21   put birth mmddyy8.;
22   put birth mmddyy6.;
23   put birth weekdate15.;
24   put birth weekdate29.;
25   run;

19JUL84
19JUL1984
19
   19
07/19/84
071984
Thu, Jul 19, 84
    Thursday, July 19, 1984
```

- SAS의 날짜기준 : 1960년1월1일 (= 0)

여러가지 날짜포맷

20050316	yymmddn8.
050316	yymmdd6.
05/03/16	yymmdds8.
05-03-16	yymmdd8.
05 03 16	yymmddb8.
2005-03-16	yymmdd10.

예6.8 : 날짜변수

```
DATA dept;
  INPUT name $6. +1 bdate DATE7. +1 hired MMDDYY8.;
  hiredate = hired + (365.25 * 3);
  hireqtr = QTR(hiredate);
  IF hired > '01jan94'D THEN new='YES';
  FORMAT bdate MMDDYY8. Hired YYMMDD8.
         Hiredate WEEKDATE17.;
  CARDS;
김철수 01jan60 09-15-90
최민지 05oct49 01-24-92
이영희 18mar88 10-10-93
오인수 29feb80 05-29-94
  ;
RUN;
```

OBS	name	bdate	hired	hiredate	hireqtr	new
1	김철수	01/01/60	90-09-15	Tue, Sep 14, 1993	3	
2	최민지	10/05/49	92-01-24	Mon, Jan 23, 1995	1	
3	이영희	03/18/88	93-10-10	Wed, Oct 9, 1996	4	
4	오인수	02/29/80	94-05-29	Wed, May 28, 1997	2	YES

5. RETAIN 명령문

- ✓SAS에서는 자료를 읽을 때 이미 주어진 자료가 없는 경우 연산을 통하여 자료를 생성해 나갈 수 있다.
- ✓이 때 연산이 이루어지기 이전에 자료값에는 default로 결측치가 들어간다.
- ✓RETAIN 명령문을 이용하게 되면 자료값을 가장 최근의 값으로 대체하게 된다.
- ✓특정 변수내에서 개체단위 연산을 수행하고자 할 때 사용함.

Example

```
DATA test;
  INPUT x @@;
  RETAIN sumx 0;
  sumx = SUM(x, sumx);
CARDS;
1 2 3 4
;
```

VIEWTABLE: Work.Test

	x	sumx
1	1	1
2	2	3
3	3	6
4	4	10

RETAIN sumx 0은 sumx에 초기값을 0으로 지정해주고 sumx에 x값을 누적하는 값이다.

예6.10 : RETAIN 명령문

```
DATA one;
   INPUT name $ x @@;
   RETAIN max_x 0 y 0 oldname '***';
   max_x = MAX(max_x,x); /* 변수 X의 자룟값 중 가장 큰 값을 계속 저장 */
   y = y +x**2;          /* y는 누적 제곱합을 계산 */
   IF name=oldname THEN case='OLD';
      ELSE case = 'NEW';
   oldname=name;
   DROP oldname;      /* DROP은 데이터셋에 저장하지 않고 제거할 변수 지정 */
CARDS;
AAA 1 BBB 4 BBB 8 BBB 3
CCC 2 CCC 5 DDD 6
;
RUN;
```

	name	x	max_x	y	case
1	AAA	1	1	1	NEW
2	BBB	4	4	17	NEW
3	BBB	8	8	81	OLD
4	BBB	3	8	90	OLD
5	CCC	2	8	94	NEW
6	CCC	5	8	119	OLD
7	DDD	6	8	155	NEW

6. ARRAY 명령문

✓ARRAY 명령문은 일련의 변수들을 배열의 원소들로 지정하는 작업을 하며, 여러 개의 변수들에 대해서 동일한 작업을 반복할 필요가 있을 때 사용된다.

ARRAY ***array명 [차원수(n)] [$] (문자길이) [변수들 이름]***

✓EXAMPLE: 5점 리커트 척도로 된 설문지의 문항 x1 ~ x6 가운데 x1, x3, x6에 대하여 역코딩을 하여야 하는 경우

```
DATA score;
  INPUT name $ x1 - x6;
  ARRAY aaa [3] x1 x3 x6;
  DO i = 1 TO 3;
     aaa[i]=6-aaa[i];
  END;
  DROP i;
CARDS;
김철수 5 5 1 2 1 2
최민지 5 3 1 4 5 3
이영희 1 5 3 2 5 4
오인수 4 1 2 4 5 1
;
RUN;
```

VIEWTABLE: Work.Score

	name	x1	x2	x3	x4	x5	x6
1	김철수	1	5	5	2	1	4
2	최민지	1	3	5	4	5	3
3	이영희	5	5	3	2	5	2
4	오인수	2	1	4	4	5	5

array

❑ array

- 여러 개의 변수명을 하나의 이름으로 사용하려면 array문을 사용하면 된다.

  array *array-name*{*#-of-variables*} <$> <*list-of-variables*>;

  여기서 array의 변수들이 문자변수면 $를 사용하고 숫자변수면 $ 없이 사용하면 된다. 이 때에 첫 번째 변수는 *array-name*{1}, 두 번째 변수는 *array-name*{2} 등의 이름으로 사용될 수 있다. 즉, buffer에서 *i* 번째 변수와 *array-name*{*i*}는 같은 기억 장소를 사용하기 때문에 *i* 번째 변수의 내용을 바꾸는 것과 *array-name*{*i*}의 내용을 바꾸는 것은 같은 효과를 가지고 있다. 그러나, buffer에서 data set으로 값이 옮겨질 때 *array-name*은 data set에 저장되지 않고 오로지 data step에서 사용된 변수명만이 data set에 저장된다.

array

예) data one;

```
    array arr{2} u v;
    input x y z;
    arr{1}=x+3;
    arr{2}=z-y;
    cards;
1 2 3
4 5 6
;
```

결과 :

u	v	x	y	z
4	1	1	2	3
7	1	4	5	6

- array문은 위와 같이 단독으로 사용되는 경우는 별로 없고 다음에 나오는 iterative do와 같이 많이 사용된다.

iterative do

❑ iterative do

- 어떠한 명령문을 반복하여 수행하려면 아래와 같이 iterative do를 사용하면 된다.

```
do index-variable = starting-value to stop-value <by increment>;
                                  :
end;
```

여기서 : 의 위치에는 SAS 문장들이 들어가는데 이 문장들이 반복하여 수행하게 된다. *increment* 가 양수일 때(*increment* 가 음수일 때) iterative do가 수행되는 방법은 다음과 같다.

iterative do

1. *index-variable*에 *starting-value*를 저장한다.
2. *index-variable*의 내용을 *stop-value*와 비교한다.
   (i) 만약 *index-variable*의 내용이 *stop-value*보다 작거나 같으면 (크거나 같으면) : 의 SAS 문장들을 수행한다.
   (ii) 만약 *index-variable*의 내용이 *stop-value*보다 크면 (작으면) end; 밖으로 나간다.
3. *index-variable*의 내용을 *increment*만큼 증가(감소)시킨다.
4. 2로 간다.

- 위에서 *increment*가 1이면 by 1을 생략할 수 있다. 즉 by가 없으면 *increment*를 1로 간주한다.

iterative do

예) data score;
array x{5} kor eng math physics chem;
input id kor eng math physics chem;
total=0;
do i=1 to 5;
total=total+x{i};
end;
drop i;
cards;
1001 93 84 74 85 84
1002 85 94 91 87 83
1003 89 93 79 84 81
1004 72 86 91 85 84
1005 97 80 93 88 91
;

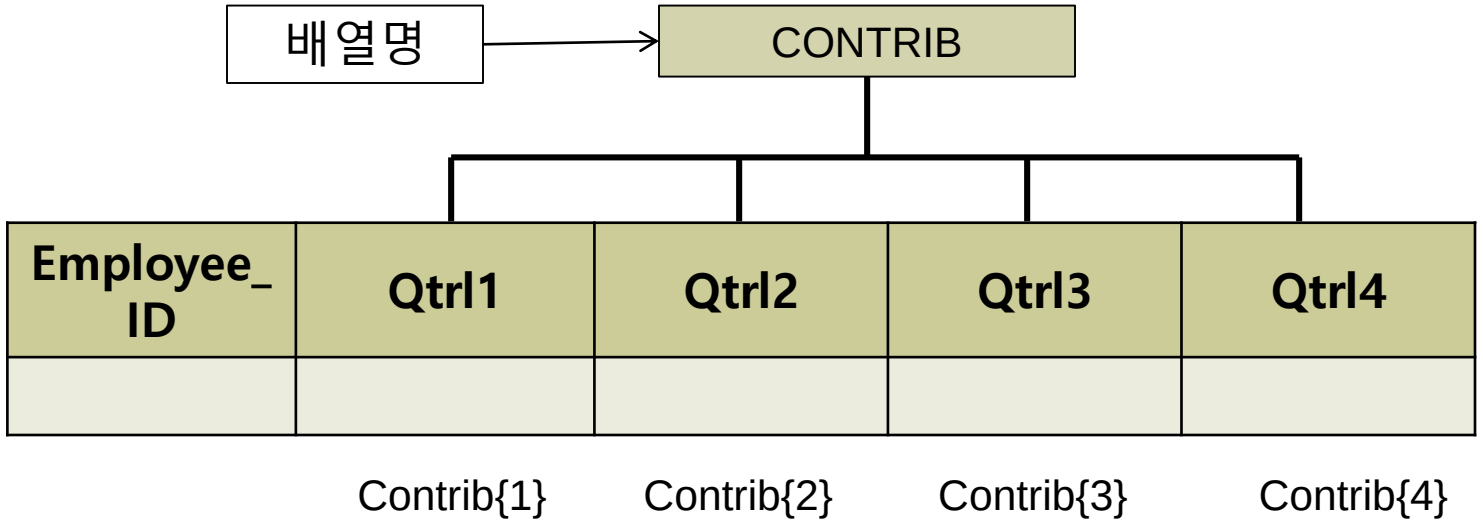
iterative do

결과 :

kor	eng	math	physics	chem	id	total
93	84	74	85	84	1001	420
85	94	91	87	83	1002	440
89	93	79	84	81	1003	426
72	86	91	85	84	1004	418
97	80	93	88	91	1005	449

ARRAY 정의

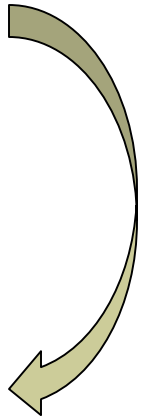
```
array Contrib{4} qtrl1 qtrl2 qtrl3 qtrl4;
```



배열과 반복문

- 반복된 계산 처리

```
DATA charity;
   set origin.employee_donation;
   keep employee_id qtrl1-qtrl4;
   Qtrl1=Qtrl1*1.25;
   Qtrl2=Qtrl2*1.25;
   Qtrl3=Qtrl3*1.25;
   Qtrl4=Qtrl4*1.25;
run;
```



```
DATA charity;
   set origin.employee_donation;
   keep employee_id qtrl1-qtrl4;
   array Contrib{4} qtrl1-qtrl4;
   do i=1 to 4;
        Contrib{i}=Contrib{i}*1.25;
    end;
run;
```

i는 1에서 4까지 수행한뒤 5에서 멈추고 2행으로 넘어가 반복수행

배열과 반복문

- 배열을 이용한 변수 생성

```
DATA change;
   set origin.employee_donations;
   Diff1=Qtrl2-Qtrl1;
   Diff2=Qtrl3-Qtrl2;
   Diff3=Qtrl4-Qtrl3;
run;
```

```
DATA change;
   set origin.employee_donations;
    drop i;
    array Contrib{4} Qtrl1-Qtrl4;
    array Diff{3};
    do i=1 to 3;
       Diff{i}=Contrib{i+1}-Contrib{i};
    end;
run;
```

Diff1~ Diff3 변수생성

인덱스변수에 대한 수식 지정 가능

ARRAY 명령문 예제(6.11)

```
DATA tscore;
  INPUT name $ math stat eng kor art;
  ARRAY tscore [5] math stat eng kor art;  /* tscore는 변수들 배열의
                                              이름을 지정 */
  DO i = 1 TO 5;
     IF tscore(i)=9 THEN tscore(i)=.;    /* ( ) [  ] 상관없음 */
  END;
CARDS;
김철수 5 5 1 2 1
최민지 9 3 1 4 5
이영희 1 5 3 2 9
오인수 4 1 2 4 9
;
RUN;
```

1. ARRAY 없이 9를 결측값으로 변경하는 방법
   IF math=9 THEN math=.;
   IF stat=9 THEN stat=.;
   IF eng=9 THEN eng=.; 등등

2. 결과를 보면 i라는 변수는 잠시 사용. 따라서 drop

	name	math	stat	eng	kor	art	i
1	김철수	5	5	1	2	1	6
2	최민지	.	3	1	4	5	6
3	이영희	1	5	3	2	.	6
4	오인수	4	1	2	4	.	6

7. 기타 명령문들(강제분기 처리 구문)

KEYWORD variable-expressions ;

사용 예	내 용
LIST ;	로그창에 현재의 데이터를 출력하라
RETURN ;	지금 까지 진행한 데이터만 저장하고, 다음 데이터 프로세스를 진행해라
GOTO label ;	label 로 가서 이후 작업을 진행해라
LINK label ;	label 에 있는 작업을 수행하고 와서 진행해라.
STOP ;	데이터 단계를 종료시키고 현재까지의 데이터로 데이타셋 생성
ABORT ;	데이터셋을 만들지 않고 데이터 단계를 종료시킴