프로그램 편람

(PL/I)

경 제 기 획 원
조 사 통 계 국

머 리 말

이 책은 IBM 조직/360 원반과 테이프 운영조직에서 PL/1 D-수준 편성자(D-水準 編成者 D-level compiler)를 사용해서 편성되는 PL/1 서브세트 프로그램(PL/1 subset program)을 쓰는 규칙을 제공한다. 이는 D-수준 편성자의 제4판에 의해(the forth version of the D-level compiler에 의해) 실행(実行)되는 PL/1 언어(言語)의 설비(設備)에 관한 참고이다.

편성과 프로그램(program)을 실행하는데 필요한 지식(知識)을 얻으려면 다음 책을 참조하라.

IBM System/360 Dish and Tape Operating Systems, PL/1 Programmer's Guide, Order NO.GC24-9005.

그 밖의 책을 보려거든 다음 책을 참조하라.

IBM System/360 Bibliography, Order NO.GA22-6822.

이 책에 있는 것은 DOS 개정 25 (DOS 改正 25 DOS realease 25)이다.

차례에는 Ⅰ부와 Ⅱ부의 차례가 전부 실려 있음.

IBM 조직 참고 총서 (組織 參考 叢書 systems reference library)

IBM 조직 / 360 (組織 / 360 IBM system/360)

원반과 테이프 운영 조직 (円盤과 테이프 運営 組織 Disk and Tape Operating Systems)

PL/1 Subset 참고서 (参考書 PL/1 Subset Reference Manual)

제 I 부

차 례

서 문

PL/1 서브세트 언어는 작은 용량(容量)의 자료 처리 조직에서 사용하도록 되어 있다. 이 서브세트는 자체독립이다. 이 말은 프로그래머가 모체인 PL/1 언어를 참고하지 않아도 이를 배울 수 있고 이용할 수 있다는 것이다. PL/1은 다른 어떤 프로그램보다 기계에 대한 종속성이 적다. 그러나 완전한 독립성이 너무 낭비가 되는 경우는 기계에 대한 종속성을 용인하고 있다.

이 책의 이용

이 책은 PL/1 프로그래머를 위한 참고서로 만들어졌다. 이 책은 두 부문으로 나뉘어, 이용자의 필요에 따라 빨리 찾을 수 있게 되어 있다. 즉 제Ⅰ부와 제Ⅱ부로 나뉘어 있다. 이 책은 현재의 D-편성자(D-compiler, D-編成者) 설비에 맞추었다. 이 책은 IBM 조직/360 원반과 테이프 운영 조직 PL/1 프로그래머 안내서와 연관되어 사용되도록 기획되었다. (IBM system/360 Disb and Tape Operating Systems PL/1 Programmer's Guide, Order NO.GC24-9005)

또 다른 책, IBM 조직/360 PL/1 참고서(IBM System/360 PL/1 Reference Manual, Order NO.GC28-8201)는 IBM 조직/360 운영 조직(IBM System/360 Operating System)에서 사용되는 F-수준 편성자(F-水準 編成者 F-level compiler)에 적용되는 이용 지식을 제공한다.

관계 서적

다음은 IBM에서 발간한 책을 소개한다.

편성, 연쇄편집 (link-edit), 프로그램을 실행하는데 필요한 지식에 대하여는, 독자는 IBM 조직/360 원반과 테이프 운영 조직 PL/1 프로그래머 안내서, order NO.GC-24-9005에 대하여 잘 알고 있어야 한다.

다음 책은 PL/1 을 배우는 이에게 유익한 다른 지식을 포함하고 있다.

PL/1 기초, order NO. GC28-6808 (A PL/1 primer,order NO. GC28-6808).

상업 프로그래머를 위한 PL/1 안내, order NO. GC20-1651 (A guide to PL/1 for commercial programmers,order NO. GC20-1651)

FORTRAN 사용자를 위한 PL/1 안내, order NO. GC20-1637 (A guide to PL/1 for FORTRAN users, order NO.GC20-1637)

제 I 부

PL/I의 개념(概念)

제 1 장 PL/I의 기본특성 (基本特性)

이 장은 언어 전반에 걸치는 것을 약술한다.

1. 기계의 독립성

어느 언어도 완전히 기계와 독립적일 수 없으나, 다른 어느 언어보다는 더 독립적이다. 이러기 위해서, 어느 면에서는 댓가를 많이 치루고 있다.

2. 프로그램 구조 (構造 program structure)

하나의 PL/I 프로그램은 하나 또는 하나 이상의 수속 (手続 procedure) 이라 하는 문 (文 statement) 의 뭉치로 이루어진다.

하나의 수속은 주 프로그램 (主 program) 또는 버금과정 (버금過程 subroutine) 이 된다. 수속은 다른 수속을 부를 수도 있고, 이들 수속이나 버금과정은 각기 따로 편성되거나, 부르는 수속에 들어있어서 함께 편성될 수도 있다.

3. 자료 형식과 자료 기술 (資料 形式과 資料 記述)

PL/I의 특성은 자료 형식의 다양성 (多様性) 에도 있다. PL/I은 산수자료, 줄 자료, 프로그램 통제 자료를 취급한다. 산수 자료는 여러가지 방법으로 표시된다 ; 이는 이진수 또는 십진수 고정점수 또는 부동점수가 될 수 있고, 이의 정도 (精度 precision) 는 지정될 수도 있다.

PL/I 은 산수 연산, 비교 연산, 비트 줄의 논리(論理) 취급, 문자 줄을 모으고 검사하고 쪼개는 연산과 기능을 수행하는 설비를 가지고 있다.

4. 태만 대행(怠慢 代行 default assumptions)

PL/I의 중요한 설비의 하나는 태만 해석이다. 만약 이름과 연관(聯関)된 모든 속성, 또는 한개의 문에 허용된 모든 선택항이 프로그래머에 의해 지정되지 않으면, 속성이나 선택항은 편성자에 의해 제공된다. 이 태만시 행위는 두가지 중요한 결과를 낳는다. 첫째 이는 요구되는 선언과 다른 프로그램을 쓰는 수고를 덜어준다; 둘째 이는 프로그래머가 모든 것을 다 알지 않아도 언어를 가르치고 사용할 수 있게 해 준다.

5. 기억소 할당(記憶所 割当 storage allocation)

PL/I은 다른 언어보다 기억소 할당의 유연성에서 앞지르고 있다. 세가지 기억소 종류가 있다: AUTOMATIC, STATIC, BASED. 보통, 태만 기억소 종류는 AUTOMATIC이다.

6. 식(式 expressions)

PL/I에서 계산은 식으로 지정된다. PL/I에서 식은 대수의 그것과 같다.

보기:

A + B * C

윗 식은 A더하기 (B * C)이다.

복합 식이 지정되면, 연산항이 그 연산이 유의한 값이 나올 수 있게끔 변환된다. 그러나 변환 규칙은 조심스레 검토되어야 한다. 변환된 자료는 원래의 자료와 다른 값을 가질지도 모른다. 어떤 변환도 편성자가 생산한 추가되는 명령을 필요로 한다. 이는 실행 시간을 길게한다.

식의 값낸 결과는 대입문(代入文)으로 변수(変数 variable)에 대입된다.

X = A + B * C ;

7. 자료 모듬(data collections)

PL/I은 자료를 모아서 기술하거나 연산을 하는 법을 허용한다. 배열은 하나 이상의 차원(次元 dimension)을 가진 나열이나 표로 모아진, 같은 형식의 자료 집합체이다. 구조체(構造体 structure)는 자료의 계급적 집합체이며, 같은 형식을 갖지 않아도 된다. 각 수준의 계급은 더 낮은 수준의 구조체를 가질 수도 있다. 가장 낮은 수준의 계급은 요소 자료 항목이나 배열이 된다.

배열은 구조체를 갖지 못하나 구조체는 배열을 가질 수 있다. 연산은 배열, 구조체 또는 배열이나 구조체의 일부에 대해 지정될 수 있다.

8. 입력과 출력(入力과 出力 input and output)

입력과 출력에 대한 설비는 평이성, 기계 독립성, 효율들의 요인중에서 선택할 수 있게끔 허용한다. PL/I에서 두가지 커다란

입력과 출력 방식이 있다. 그것은 줄-지향(줄-指向 stream-oriented)과 기록마디-지향(記錄마디-指向 record-oriented)이다.

줄-지향 입력/출력은 거의 완전히 기계와의 독립성이다.

기록마디 입력/출력은 기계에 종속이다.

9. 중단 활동(中斷 活動 interrupt activities)

최신의 계산 조직은 예외 조건이 일어났을 때 프로그램의 실행을 중단하는 설비를 갖추고 있다. 한걸음 나가서, 그것은 예외 조건을 다루고 중단이 일어난 점으로 돌아가게 한다.

PL/I은 여러가지의 예외 조건을 찾아내는 설비를 갖추고 있다. 이는 조건이 일어나면 중단이 일어날것인가 아닌가를 조건 전치어의 수단으로 지정할 수 있게 한다. 또, ON문을 사용해서, 프로그래머는 중단이 일어날 때 취하는 행위를 지정할 수 있다.

外
1
민

제 2장 프로그램 요소 (要素 program elements)

PL/I 문 (文 statements) 의 서식에는 제약이 적다. 그런 결과로, 프로그램이 정해진 부호 형이나 각 문이 정해진 자리에서 시작했나를 검사할 필요없이 써질 수 있다. 각개의 문이 세미콜론 (;) 으로 끝나면, 그 서식은 자유이다. 각 문은 앞 문의 다음 자리부터 시작할 수 있고 또 몇개의 빈자가 중간에 있어도 좋다. D - 편성자는 원시 프로그램에서 매 카드의 첫째 자리는 빈자여야 한다 ; 이 카드의 73자리부터 80자리는 무시되고 다른 어떤 참고 사항을 넣을 수 있다.

이상을 정리하면,

〔1〕 제1자리는 빈자 (blank)

〔2〕 제2～제72자리는 프로그램 문, 여기에는 주 (註) 도 포함한다.

〔3〕 제73～제80자리는 참고란. 이 난은 프로그램 실행과 관계없음. 즉 주 (註) 와 같음.

〔4〕 문은 세미콜론으로 끝나야 한다.

〔5〕 문과 문사이에 하나 이상의 빈자를 두어도 좋다.

더 상세한 것은 앞으로 나올 것이다.

제 1절 문자 조 (文字 組 character set)

두개 문자 조중의 하나가 원시 프로그램을 쓰는데 사용될 수 있다 ; 60 - 문자 조나 48 - 문자 조 중에 하나이다. 주어진 외부 수속 (外部 手続 external procedure) 에서, 두개의 조 중에서

택하는 것은 자유이다. 실제로, 이 선택은 사용되는 장비에 따라 다를 것이다.

1. 60-문자 조 (60-文字 組 60-character set)

60-문자 조는 숫자, 특수 문자, 영문자로 되어 있다.

〔1〕 영문자 (또는 영자 英字 alphabetic characters)

$, #, @, A, B, ... Y, Z (29자)

* $, #, @가 포함됨을 유의하라.

〔2〕 숫자 (digits 또는 numeric characters)

0, 1, 2, ... 8, 9 (10자)

〔3〕 특수 문자 (特殊 文字 special characters)

특수 문자에는 21개가 있으며 다음과 같다. 편의상 영어로도 이름을 달았다.

문 자	이 름
=	등호 또는 대입 부호 (等号, 代入符号 equal or assignment symbol)
+	양부호 (陽符号 plus sign)
—	음부호 (陰符号 minus sign)
*	별표 또는 곱하기 부호 (asterisk or multiply symbol)
/	빗금 또는 나누기 부호 (slash or divide symbol)
(	왼쪽 괄호 (括弧 left parenthesis)
)	오른쪽 괄호 (right parenthesis)
,	쉼표 (comma)
.	마침표 또는 소수점 (period or point)

'	반따옴표 또는 생략 부호(apastrophe)
%	백분 기호(百分 記号 precent symbol)
;	세미콜론(semicolon)
:	콜론(colon)
¬	// 아니다 // 부호(// Not // symbol)
&	// 그리고 // 부호(// and // symbol)
!	// 또는 // 부호(// or // symbol)
>	// ~보다크다 // 부호(// greater than // symbol)
<	// ~보다작다 // 부호(// less than // symbol)
—	절단 문자(break character)
?	물음표(question mark)

* 절단 문자는 음부호와 다름을 유의하라.

이의 보기 :

GROSS-PAY, NET-PAY 등과 같이 쓰면 좋다.

특수 문자 중에 어떤 것은 다른 것과 결합해서 특별한 뜻으로 쓰기도 한다.

보기 :

<=, ¬=, ¬< 등

영수 문자(또는 영수 자 alphameric character)는 영자이거나 숫자이다.

2. 48 -문자 조(48-character set)

48 -문자 조는 60 -문자 조 중에서 48개 문자로 이루어진다. 축소된 조의 문자는 큰 조에 없는 문자를 나타내기 위해 결합되기도 한다. 보기로, 세미콜론(;)은 48 -문자 조에 없으

나, 쉼표와 마침표를 빈자 안 넣고 이어서 이를 대신할 수 있다; 즉 ,.로 쓴다.

〔1〕 영자

$, A, B……Y, Z (27자)

〔2〕 숫자

0, 1, 2, ……8, 9 (10자)

〔3〕 특수 문자

b, ., +, -, *, (,), /, ,, ', = (11자)

상세한 것은 "제Ⅱ부 제2장 EBCDIC와 카-드 천공 규약의 문자 조"를 보라.

3. 문자 조의 이용

PL/I 프로그램을 구성하는 모든 요소는 PL/I 문자 조로 부터 만들어진다. 여기에 두 가지 예외가 있다: 그것은 문자 줄 상수와 주석은 그 기계에서 허용되는 어떤 문자로도 이루어질 수 있다.

어떤 문자는 PL/I 프로그램에서 지정된 역할을 한다.

〔1〕 연산자(演算子 operators)

연산자에는 네가지 종류가 있다.

1〕 산수 연산자(算数 演算子 arithmatic operators)

+ 더하기 또는 양부호(addition or prefix plus)

- 빼기 또는 음부호(subtraction or prefix minus)

* 곱하기(multplication)

/ 나누기(division)

** 제곱(exponentiation)

표 2 - 1 특수 문자의 역활

이 름	문 자	사 용
쉼 표	,	나열의 요소를 분리.
마 침 표	.	2진 또는 십진의 소숫점; 수식된 이름의 요소 연결.
세 미 콜 론	;	문을 끝냄.
대 입 부 호	=	값의 대입
콜 론	:	문에 전치어를 연결
빈 자		문의 요소를 분리
생 략 기 호	,	줄 상수와 모양 지정을 싼다.
괄 호	( )	나열을 싼다. 여러가지 열쇠와 연관된 정보를 지정. 연산자와 연산항과 관련해서 계산식의 부분을 구분

문자=는 등호와 대입 부호로 쓴다.

AB + BC	는	AB + BC	와 같다.
TABLE(10)	는	TABLE(10)	와 같다.
FIRST,SECOND	는	FIRST, SECOND	와 같다.
ATOB	는	A TO B	와 같지않다.

도 해 2 - 1 빈자 사용의 보기

2] 비교 연산자 (比較 演算子 comparison operators)

> 〃보다 크다〃 (〃 greater than 〃)

¬> 〃보다 안 크다〃 (〃 not greater than 〃)

>= 〃보다 크거나 같다〃 (〃 greater than or equal to〃)

= 〃과 같다〃 (〃 equal to 〃)

¬= 〃과 안 같다〃 (not equal to 〃)

<= 〃보다 작거나 같다〃 (〃 less than or equal to 〃)

< 〃보다 작다〃 (〃 less than 〃)

¬< 〃보다 안 작다〃 (〃 not less than 〃)

3] 비트줄 연산자 (bit-string operators)

¬ 〃아니다〃 (〃 not 〃)

& 〃그리고〃 (〃 and 〃)

| 〃또는〃 (〃 or 〃)

4] 줄 연산자 (string operato)

|| 잇기 (concatenation)

표 2 - 1은 다른 특수 문자의 역할을 보여준다.

〔2〕 구분자 (区分子 delimiters)

두 종류의 구분자가 있다 : 그것은 연산자와 분리자이다. (operators and separators) 도해 2 - 1에서 등호와 생략기호를 빼고는 분리자이다.

〔3〕 표식어 (標識語 identifiers)

프로그램에서 이름이나 명찰은 자료, 화일, 문, 다른 프로그램 영역의 입구점들에 붙을 수 있다. 이름이나 명찰을 만드는데 있어서 프로그래머는 표식어를 만드는 구문 규칙을 파악해야

한다.

표식어는 한개의 영자거나 31자 이내의 영수자와 절단 문자 줄이다. 첫 문자는 영자이어야 한다.

언어의 열쇠말(keywords)도 표식어이다. 열쇠말이란 특정 문맥에서 사용되면 편성자(compiler)에 대해 정해진 뜻을 가지는 것이다. 보기로 READ, DECIMAL, ENDFILE 들이다.

어떤 열쇠말은 줄일 수 있다. 이 준말은 그 자체로 열쇠말이 되며 완전한 열쇠말과 모든 면에서 같은 뜻으로 인지된다. 열쇠말과 이의 준말은 제Ⅱ부 제3장 〃열쇠말과 열쇠말 준말〃을 보라.

주 : 대부분의 PL/I 열쇠말은 예약된 말이 아니다. 그들은 특정한 문맥(文脈)에 나타날 때만 열쇠말로서 인지된다. 다른 문맥에서는 그들은 프로그래머가 정의한 표식어로서 사용되어도 된다. (예약된 열쇠말은 제7장 〃이름의 인지〃를 보라.)

표식어는 31자를 넘지 못한다. D-편성자에서 뒷장에서 기술하는 어떤 표식어는 6개 문자를 넘지 못한다. 이 제한은 외부이름이라하는 특정 이름에만 가해진다.

표식어의 보기 :

A

FILEZ

Loop-3

#32

〔4〕 빈자의 사용

빈자는 PL/I 프로그램 전반에서 자유롭게 사용될 수 있

다. 일반으로 하나의 빈자를 허용하는 곳은 몇개의 빈자가 있어도 좋다.

하나 또는 그 이상의 빈자가 표식어와 다른 구분자나 주석에 의해 분리되지 않은 상수를 분리해야 한다. 그러나 표식어, 상수(문자 줄 상수는 제외), 복합 연산자(보기, ㄱ=)는 빈자를 포함하면 안된다.

빈자를 요구하거나 허용하는 경우는 언어의 구조가 기술될 때 설명될 것이다. 보기로 도해 2-1을 보라.

〔5〕 주석(註 comments)

주석은 문자-줄 상수와 같은 자료 항목 안에서를 제외하고는 빈자가 허용되는 어떤 곳에서도 허용된다. 주석은 빈자처럼 취급되고 따라서 분리하는 빈자 대신 사용되어도 된다. 주석은 달리 프로그램의 실행에 영향을 미치지 않는다. 주석은 문과 같은 카-드에 천공해도, 문과 문 사이에 또는 문의 중간에 끼워도 된다.

주석의 서식:

/* 문자줄 */

문자의 짝 /*은 주석의 처음을, 짝 */은 마지막을 가리킨다. 이 짝의 가운데 빈자를 끼울 수 없다. 주석은 */ 외에 어떤 문자로도 이루어 질 수 있다.

보기:

```
/*  THIS IS COMMENT  #5  */
/*  /*  */
/* * * / # # # * /
```

제2절 기본 프로그램 구조

PL/I 프로그램은 문(文 statements)이라 하는 기본 프로그램 요소로써 만들어진다.

두 종류의 문이 있다 : 단순문(simple)과 복합문(compound)

1. 단순문과 복합문(simple and compound statements)

세가지 종류의 단순문이 있다 : 열쇠말(keyword), 대입(代入 assignment), 빈(null)문이다. 각각의 문은 세미콜론으로 끝나는 문 몸체가 있다.

〔1〕 열쇠말 문(keyword statement)

열쇠말 문은 문의 역할을 나타내는 열쇠말을 가진다 ; 문의 몸체는 문의 나머지가 된다.

〔2〕 대입 문(代入 文 assignment statement)

대입 문은 대입기호(또는 등호)(=)가 있고 열쇠말이 없다.

〔3〕 빈 문(null statement)

빈 문은 하나의 세미콜론만이 있고 연산이 없음을 나타낸다 ; 세미콜론이 문 몸체가 된다.

보기 :

GOTO LOOP3 ; (GOTO는 열쇠말이다 ; 문 몸체는 LOOP3이다.)

A = B + C ; (대입 문)

〔4〕 복합 문(複合 文 compound statement)

복합 문은 문 몸체의 부분으로 하나 이상의 다른 문을 가지고 있는 문이다. 두 개의 복합 문이 있다 : IF문과 ON문

이다. 복합 문의 마지막 문은 단순 문이다.

보기 :

1. IF A>B THEN A = B + C ; ELSE GO TO LOOP3 ;

2. ON UNDERFLOW GO TO UNFIX ;

3. ON UNDERFLOW ;

보기 3은 빈 문(null statement)이다.

2. 문 전치어(文 前置語 statement prefixes)

단순과 복합문은 하나 이상의 전치어(prefix)를 가질 수 있다. 두 가지의 전치어가 있다 : 명찰 전치어(名札 前置語 label prefix)와 조건 전치어(条件 前置語 condition prefix)이다.

〔1〕 명찰 전치어

명찰 전치어는 문이 다른 어떤 점에서 인용될 수 있도록 문을 동일시한다.

명찰 전치어는 문앞에 오는 하나의 표식어이며 콜론(:)으로 문에 연결된다. 하나 이상의 명찰을 대부분의 문은 가질 수 있다. 두개 이상이면, 그 문을 인용하기 위해 아무거나 쓸 수 있다. PROCEDURE와 ENTRY문은 한개의 명찰만 가져야 한다.

〔2〕 조건 전치어(条件 前置語 condition prefix)

조건 전치어는 중단이 지명된 조건의 출현으로 일어날 것인가를 지정한다. 조건 이름은 열쇠말이다. 각개는 프로그램의 실행 도중 일어날지도 모르는 예외 조건을 표시한다. 보기로 OVERFLOW SIZE.

조건 이름은 조건이 일어나도 중단이 일어나지 말라고 앞에 NO란 말이 붙을 수 있다. 조건 전치어는 하나 이상의 조건 이름

의 나열로 될 수 있으며 쉼표로 구분되고 괄호에 싸인다. 조건 전치어는 문의 제일 앞에 오며 명찰이 붙은 경우도 같다.

보기 :

(SIZE,NOOVERFLOW) : COMPUTE : A = B * C * * D ;

또 보기에 좋게 쓴다면

(SIZE,NOOVERFLOW) :
COMPUTE : A = B * C * * D ;

3. 모임과 블러크(groups and blocks)

〔1〕 모임(group)

모임이란 DO로 시작해서 대응하는 END로 끝나는 문의 연속이다. 이는 통제의 목적으로 사용된다. 모임은 DO-모임이라고도 한다.

〔2〕 블러크(block)

블러크는 프로그램의 한 영역을 정의한 문의 연속이다. 이는 이름의 범위를 구분하며 통제 목적으로 사용된다. 프로그램은 하나 이상의 블러크로 이루어진다. 모든 문은 블러크 안에 있어야한다. 두 종류의 블러크가 있다 : 시작 블러크(begin block)와 수속 블러크(手続 procedure block)이다. 시작 블러크는 BEGIN 문과 END문으로 구분된다. 모든 시작 블러크는 다른 블러크 안에 들어야 한다.

실행은 순차로 시작 블러크로 들어가서 밖으로 나온다. 그러나 수속 블러크는 다른 블러크의 문에 의해 불러내질 수 있다. 한 개의 프로그램에서 첫번째 실행될 수속은(때로는 주(主 main) 또는 초기 (初期 initial)수속이라고도 함) 운영 조직(operating system)에 의해 자동으로 불러내진다. 조직/360에서 이 첫번째 수속은 PROCEDURE 문에서 OPTIONS(MAIN)을 지정하므로써 정해진다.

제 3 장 자료 요소 (資料 要素 data elements)

자료 (資料 data) 란 일반으로 정보 (또는 지식) 나 값의 (情報 (또는 知識) 나 값의 (information or value)) 표현 (表現) 으로 정의된다.

PL/I에서, 자료를 인용함은 산수든 줄이든, 변수 (変数 variable) 나 상수 (常数 constant) 를 사용해서 달성된다 (변수, 상수란 용어는 보통 수학상 사용과 꼭 같지는 않다.)

변수란 프로그램의 실행도중에 바뀌질 수 있는 값을 가지는 상징적 이름 (symbolic name) 이다.

상수란 (이는 상징적 이름이 아니다.) 바꾸지 못하는 값을 가진다.

다음 문은 변수와 상수를 다 함께 가지고 있다.

```
AREA = RADIUS ** 2 * 3.1416 ;
 ABC = ' ABC ' ;
```

제 1 절 자료 형식 (資料 形式 data types)

PL/I 프로그램에 사용되는 자료 형식은 두개의 범주 (範疇) 로 떨어진다 : 문제 자료 (問題 資料 problem data) 와 프로그램 통제 자료 (프로그램 統制 資料 program control data) 이다.

문제 자료는 프로그램에 의해 처리되는 값을 나타낸다. 이는 산수와 줄 자료 형식 (算数와 줄 資料 形式 arithmatic and string data types) 으로 나뉜다.

프로그램 통제 자료는 프로그램의 실행을 통제하기 위하여 사용된다. 문 명찰과 지침 (指針 pointer) 은 프로그램 통제 자료

형식이다.

상수는 값을 잘 말해준다. 이는 자료 항목의 여러가지 특성을 잘 설명해 준다. 보기로, 3.1416은 자료 형식은 산수(arithmatic)이며 자료 항목은 다섯자리의 십진수이며 이 숫자중의 네 자리는 소수점 오른편에 있음을 보여 준다.

변수의 특성은 이름만으로서는 명백치 않다. 속성이라 하는 이들의 특성이 알려져야 하므로, 어떤 열쇠말과 식이 DECLARE 문에서 이 변수의 속성을 지정하기 위하여 사용될 수 있다. 각개의 자료 형식을 기술하는 속성이 이 장에서 간단히 논술된다. 각 속성의 완전한 논술은 제Ⅱ부 제9장, "속성"에 있다.

제2절 문제 자료(問題 資料 problem data)

문제 자료의 형식은 산수와 줄(算数와 줄 arithmatic and string)이다.

1. 산수 자료(算数 資料 arithmatic data)

산수 자료의 항목은 수치 값을 가진 것이다. 산수 자료 항목은 기수(基数 base), 척도(尺度 scale), 정도(精度 precision)의 특성을 가진다. 산수 변수에 의해 표현된 자료 항목의 특성은 그 이름을 위해 선언된 또는 태만에 의해 대행(代行)된 속성에 의해 지정된다.

산수 자료 항목의 기수(基数 base)는 십진(十進 decimal) 또는 이진(二進 binary)이다.

척도(尺度 scale)는 고정점수(固定点数 fixed-point) 또는 부동점수(浮動点数 floating-point)이다.

정도(精度 precision)는 고정점수에서 자료 항목이 가질 수 있는 자리수이며, 부동점수에서 최소의 유의(有意) 숫자(지수도 넣어서) 자리수이다. 십진 고정점수 자료 항목에서, 정도는 숫자의 오른쪽에서 부터의 소수점 자리를 지정할 수 있다.

산수 변수의 기수와 척도는 열쇠말로 지정된다 ; 정도는 괄호로 싸인 십진 고른 상수로 지정된다.

〔1〕 십진 고정점수 자료(十進 固定点数 資料 decimal fixed-point data)

1〕 십진 고정점 상수(常数 constant)

십진 고정점 상수는 소수점이 있을 수 있는 하나 이상의 십진 숫자로 된다. 소수점이 없으면, 수의 맨 오른쪽 수의 바로 다음으로 대행된다. 음양 부호가 앞에 붙을 수 있다.

보기 :

3.1416, 732, 0.0012, •0012, +45, -25.00

2〕 십진 고정점 변수(decimal fixed-point variables)

십진 고정점 변수를 선언하는 열쇠말 속성은 DECIMAL과 FIXED이다. 정도는 괄호로 싸이고 쉼표로 분리된, 두개의 부호 없는 십진 고른 상수로 표시된다. 첫째 수는 전체 자리수, 두번째 수(척도 인자(尺度 因子 scale factor))는 소수점 오른쪽의 자리수를 지정한다.

변수가 상수만 표현한다면, 척도 인자와 쉼표가 생략될 수 있다. 속성은 어떤 순서로 나타나도 좋으나, 정도 지정은 DECIMAL이나 FIXED 다음에 와야 한다.

보기 :

```
DECLARE A FIXED DECIMAL(5,4);
```

DECLARE B FIXED(6,0) DECIMAL;

DECLARE C DECIMAL(6) FIXED;

조직/360에서 허용된 최대 자리수는 15이다. 십진 고정점수 자료의 내부 규약 산수 꼴(內部 規約 算数 꼴 internal coded arithmatic form)은 짜인 십진수(packed decimal)이다. 정도가 없으면 (5,0)로 대행된다.

〔2〕 이진 고정점수 자료(二進 固定点数 資料 binary fixed-point data)

1〕 이진 고정점 상수

이진 고정점 상수는 바로 뒤에 문자 B가 붙은 하나 이상의 이진 숫자로 된다. 이 수는 이진 소수점이 없다.

소수점은 맨 오른쪽 다음에 오는 것으로 대행된다. 부호가 앞에 올 수 있다.

보기:

10110B, 1111B, +101B, -1101B

2〕 이진 고정점 변수(binary fixed-point variables)

이 수를 선언하기 위한 열쇠말 속성은 BINARY와 FIXED이다. 정도는 괄호로 싸인 십진 고른수이며, 변수가 가지는 최대 이진 수의 자리수를 표시한다. 이진 고정점수는 고른수(integer)를 표현한다. (D-편성자에서 이 수의 소수점은 없다) 속성은 어떤 순서라도 되나, 정도는 BINARY나 FIXED 뒤에 와야 한다.

보기:

DECLARE FACTOR BINARY FIXED(20);

DECLARE B BINARY(15) FIXED;

外
1
민

조직/360에서 허용된 최대 자리수는 31이다. D-변성자에서 태만 정도(怠慢 精度 default assumption)는 (15)이다. 이진 고정점수 자료의 내부 규약 산수 자료는 고정점 이진수 온말(fixed-point binary full word)이다. 한개의 온 말은 31개 비트 더하기 부호 비트이다. 이 수의 어떤 항목도 31자리로 기억된다. 비록 31자리 안되게 선언되었을지라도. 선언이 되지 않은 표식어는 첫문자가 I부터 N까지라면, 태만 정도와 함께 이진 고정점 변수로 된다.

〔3〕 영국돈 고정점수 자료(英国돈 sterling fixed-point data)

PL/I은 영국돈 값의 상수를 다루는 설비를 가지고 있다. 자료는 마침표로 분리된 파운드, 실링, 펜스 난으로 기록된다. 이 자료는 내부에서 펜스로서 등가인(等価인) 십진 고정점수로 바뀌고 처리된다. 영국돈 자료 상수는 파운드 부호를 표시하는 문자 L로서 끝난다. 세개의 난 전부가(파운드, 실링, 펜스) 영국돈 상수에 나타나야 한다. 펜스난은 소수점이 있어도 좋은 하나 이상의 십진수 자리이다(고른수 부분이 12보다 작아야 하며 적어도 한개 자리는 되어야 한다)

실링난은 두 자리를 넘지 못한다.

보기 :

101.13.8L, 1.10.0L, 0.0.2.5L, 2.4.6L

세번째는 2펜스 반을 나타낸다. 마지막은 2파운드, 4실링, 6펜스를 표시한다. 이는 내부에서 534(펜스)로 바꿔지고 기억된다.

영국돈 변수를 선언하는 열쇠말 속성은 없으나, 변수는 영국돈

모양(模様)으로 선언될 수 있다. 영국돈 상수의 정도는 이것의 펜스로 표시된 값의 정도이다.

〔4〕 십진 부동점수 자료(十進 浮動点数 資料 decimal floating-point data)

1〕 십진 부동점 상수(constant)

십진 부동점 상수는 문자 E가 뒤에 붙은 십진수의 난파, 음양 부호가 붙을 수 있고 십의 제곱을 지정하는 십진 고른수 지수(指数 exponent)가 뒤따르는 것으로서 기록된다. 첫번째 난은 한개의 소수점이 있어도 된다. 상수앞에 음양 부호가 붙어도 된다.

보기 :

15E-23, 15E23, 438E0

3141593E-6, .003141593E3, -0.05E63

2〕 십진 부동점수 변수(decimal floating-point variables)

십진 부동점수 변수를 선언하는 열쇠말 속성은 DECIMAL과 FLOAT이다. 정도는 괄호로 싸인 십진 고른수 상수로 표시된다. 이는 유지되는 유의 숫자(有意 数字)의 최소 갯수를 지정한다. 변수에 대입된 항목이 선언된 변수의 정도보다 크면, 오른 편에서 잘라진다. 속성은 어떤 순서로 나타나도 좋으나, 정도 지정은 DECIMAL 또는 FLOAT 뒤에 와야 한다.

보기 :

DECLARE A DECIMAL FLOAT(5), B DECIMAL(10) FLOAT ;

조직/360에서 허용된 최대 정도는 (16)이다 ; 지수는 두 자리를 넘지 못한다. 이 수로 대략 10^{-78} 부터 10^{75} 까지 표시된

다. 태만 정도는 (6)이다. 이 수의 내부 규약 산수 꼴은 정상 십육진 부동점수(正常 十六進 浮動点数 normalized hexadecimal floating-point)이다. 선언된 정도가 (6)보다 작거나 같으면, 짧은 부동점수 꼴이 사용되고; 선언된 정도가 (6)보다 크면 긴 부동점수 꼴(long floating-point form)이 사용된다.

선언이 없는 표식어는 만약 첫 문자가 I부터 N 까지 이외의 문자이면 십진 부동점수 변수로 대행된다.

〔5〕 이진 부동점수 자료(binary floating-point data)

1〕 상수

이진 부동점수 상수는 문자 E가 따르는 이진수 난과, 문자 B가 따르는 음양 부호가 있어도 좋은 십진 고른수난이 뒤따르는 것으로 되어 있다. 지수는 십진 자리수의 연속이며 2의 제곱을 지정한다. 상수는 앞에 음양 부호가 붙을 수 있다.

보기:

101101E5B, 101.101E+2B, 11101E-28B

2〕 변 수

이 수를 선언하는 열쇠말 속성은 BINARY와 FLOAT이다. 정도는 괄호로 싸인 십진 고른수 상수로 표시되며 유지될 유의 숫자(significant digit)의 최소 갯수를 지정한다. 속성은 어떤 순서로도 나타날 수 있으나, 정도 지정은 BINARY나 FLOAT 뒤에 와야 한다.

보기:

DECLARE A BINARY FLOAT(16) ;

조직/360에서 이진 부동점수 자료 항목을 위해 허용된 최대 정도는 (53)이다. 지수는 세 자리 십진수를 넘지 못한다. 약

2^{-260} 부터 2^{252} 사이의 값이 이 수에 의해 표현될 수 있다. 이 수의 내부 규약 산수 꼴은 정상 십육진 부동점수이다. 만약 선언된 정도가 (21)보다 작거나 같으면 짧은 부동점수 꼴(short floating-point form)이 사용되고, (21)보다 크면 긴 부동점수 꼴이 사용된다.

〔6〕 수치 문자 자료(数値 文字 資料 numeric character data)

수치 문자 자료 항목(수치 난 자료 항목(numeric field data item)이라고도 함) (numeric character data item)은 PICTURE 속성과 수치 모양 지정(numeric picture specification)으로 선언된 변수의 값이다. 이 자료 항목은 십진 고정점수나 부동점수 값의 문자 표현이다. (각각 15와 16의 최대 정도를 가지고)

수치 모양 지정은 산수 값이 대입될 문자의 연속을 말한다. 수치 모양 지정의 기본 꼴은 하나 이상의 모양 문자 9와 소수점의 자리를 가리키는 모양 문자 v로 된다. 모양 지정은 반따옴표(single apostrophes)로 둘러 싸여야 한다.

보기 :

'999V99'

이는 문자 꼴로서 다섯 자리 십진수이며, 오른편에서 두번째 자리 앞에 소수점을 가지는 자료 항목임을 말해준다.

반복 인수(反複 因数 repetition factors)가 수치 모양 지정에서 사용될 수 있다. 반복 인수는 괄호로 싸인 십진 고른수 상수이며, 이는 바로 뒤에 오는 모양 문자의 반복 번수(番数)를 가리킨다. 반복 인수의 길이는 세 자리로 한정된다.

보기 :

'(3)9V(2)9'

이는 앞서 보인 보기와 같은 것이다.

수치 문자 변수를 기술하는 서식은:

DECLARE 표식어 PICTURE '수치-모양-지정' ;

보기:

DECLARE PRICE PICTURE '999V99' ;

수치 문자 자료는 산수의 특성을 가지나 이는 규약 산수 꼴로 기억되지 않음을 유의해야 한다. 조직/360에서 수치 문자 자료는 구획된 십진수 형으로(区劃된 十進数 形 zoned decimal format) 기억된다;이것이 산수 계산에 사용되기 전에, 짜인 십진수나 십육진 부동점수 형으로 바뀌어야 한다. 이런 변환은 자동으로 되고, 그러나 별도의 실행시간이 든다.

편집하는 문자가 수치 문자 자료 항목에 끼워지게 지정될 수 있고, 이런 문자는 자료 항목 안에 실제로 기억된다. 또 자료 항목이 문자 줄(character string)에 대입되면, 이 편집하는 문자도 대입이 된다. 만약 수치 문자 항목이 다른 수치 문자나 산수 변수에 대입 된다면, 편집하는 문자는 대입이 안된다; 다만 실 숫자와 소수점의 자리만이 대입된다.(문자 줄 자료는 수치 문자 변수에 대입되지 못한다.)

보기:

DECLARE PRICE PICTURE '$99V.99', COST CHARACTER(6),

VALUE FIXED DECIMAL(6,2) ;

PRICE = 12.28 ; COST = '$12.28' ;

PRICE를 위한 모양 지정에서, 유동 부호($)와 소수점(.)는 편집 문자이다. 이들은 자료 항목에 문자로서 기억된다. 그렇지만, 산수 값의 일부는 아니다. 위 두 문의 실행후, 실제의 내부

문자 표현은 같다고 할 수 있다.

그러나 기능상 같지는 않다.

보기 :

```
VALUE = PRICE ;     COST = PRICE ;
VALUE = COST ;      PRICE = COST ;
```

나중의 두 문은 오류이다. 이는 문자 줄은 수치 문자로 바뀌지 않는다.

그밖에, 영 지우는 문자, 부동문자, 삽입 문자 등의 편집 문자가 수치 모양 지정에서 사용될 수 있다.

모양 문자에 대한 자세한 것은 제Ⅱ부, 제4장, "모양 지정 문자"와 제Ⅱ부, 제9장, "속성"에 있는 PICTURE 속성에 대한 논술을 보라.

2. 줄 자료 (string data)

줄이란 한개의 자료 항목으로 취급되는 문자 (또는 이진 숫자) 연속이다. 줄의 길이는 가지고 있는 문자 (또는 이진 숫자) 의 수이다. 두가지 형식의 줄이 있다 : 비트 줄 (bit string) 과 문자 줄 (文字 줄 character string).

〔1〕 문자줄 자료 (character-string data)

문자 줄은 문자로서 기계에 부착된 모든 숫자, 문자 (letter), 특수 문자를 포함한다.

1〕 문자-줄 상수 (character-string constants)

문자-줄 상수는 반 따옴표로 싸여야 한다. 한개의 반 따옴표가 줄에서 문자가 되려면, 이는 가운데 빈자가 들어가지 않은 두개의 반 따옴표로 써져야 한다. 두개의 반 따옴표가 한개

의 반 따옴표를 표시하기 위하여 줄에서 사용되면, 이는 한개의 문자로 계산된다.

보기 :

'LOGARITHM TABLE'

'PAGE 5', 'SHAKESPEARE'S ''''HAMLET'''''

'AC438-19', (2)'WALLA'

세번째 보기는 실제로 SHAKESPEARE'S "HAMLET"의 24의 길이다. 마지막 보기에서, 괄호로 싸인 수는 다음에 오는 문자의 반복을 가리키는 반복 인수(repetition factor)이다. 이 보기는 실제로 'WALLAWALLA'이다. 반복 인수는 괄호로 싸인 부호 없는 십진 고른수 상수이다. 반복 인수의 길이는 세 자리로 한정된다.

2] 문자-줄 변수(character-string variables)

문자-줄 변수를 선언하는 열쇠말 속성은 CHARACTER이다. 길이는 괄호로 싸인 십진 고른수 상수에 의해 선언되며, 이는 줄의 문자 수를 지정한다. 길이 지정은 열쇠말 CHARACTER 뒤에 와야 한다.

보기 :

DECLARE NAME CHARACTER(15) ;

조직/360에서, 문자-줄 자료는 내부에서 문자 형으로 유지되고 이는 각 문자는 기억소의 한 자(byte)를 차지한다. 허용된 최대 길이는 255이다.

3] 문자 모양(character picture)

문자-줄 변수는 PICTURE 속성을 사용해서도 선언될 수 있다.

꼴 :

PICTURE ' 문자 - 모양 - 지정 '

문자 모양 지정은 모양 지정 문자 X로 된 줄이다. X의 줄은 반 따옴표로 싸여야 한다. 문자 X는 그 난에서 어떤 문자도 나타날 수 있음을 지정한다.

보기 :

DECLARE PART PICTURE 'XXXXX' ;

이는 5개의 문자를 나타낸다.

반복 인수가 사용될 수 있다. 반복 인자는 반 따옴표 안에 있어야 한다.

보기 :

DECLARE PART PICTURE '(5)X' ;

이는 먼저의 보기와 같은 것이다.

모양 지정을 위해 허용된 최대 길이는 앞서 말한 문자 - 줄 상수를 위한 것과 같다. 즉 255.

〔2〕 비트 - 줄 자료(bit-string data)

1〕 비트 - 줄 상수(bit-string constant)

비트 - 줄 상수는 반 따옴표로 싸인 이진 숫자의 연속과 문자 B를 뒤에 붙인 것으로 표현된다.

보기 :

'1'B, '1101'B, (64)'0'B

마지막 보기에서 괄호로 싸인 숫자는 다음에 오는 숫자의 연속이 지정된 번수(番数)만큼 반복됨을 지정하는 반복 인수이다. 반복 인수는 부호 없는 십진 고른수 상수이며 괄호로 싸인다. 반복 인수의 길이는 세 자리로 한정된다.

2〕 비트 - 줄 변수는 BIT 열쇠말 속성으로 선언된다. 길이는

괄호에 싸인 십진 고른수 상수에 의해 지정되며, 이진 숫자의 를 가리킨다. 길이 지정은 열쇠말 BIT 뒤에 와야 한다.

보기 :

DECLARE SYM BIT(64) ;

문자 줄처럼, 비트 줄은 왼쪽에서 오른쪽으로 변수에 대입된다. 만약 줄이 변수에 선언된 길이보다 길면, 가장 오른쪽이 잘린다. 만약 짧으면 오른쪽이 영으로 채워진다.

조직/360에서, 비트 줄은 한 자(byte 字)에 8개 비트가 기억되며, 각 줄은 자 경계로 할당된다. D-편성자에서 비트 줄 변수에 허용된 최대 길이는 64이다. 비트 줄 상수에 허용된 길이도 64이다. 최소 길이는 두 경우 1이다.

제3절 프로그램 통제 자료(統制 資料 program control data)

프로그램 통제 자료의 형식은 명찰과 지침(名札 label과 指針 pointer)이다.

1. 명찰 자료(名札 資料 label data)

명찰 자료는 프로그램 통제 자료의 하나이다. 명찰 자료 항목은 명찰 상수(label constant)이거나 명찰 변수(label variable)의 값이다.

〔1〕 명찰 상수(名札 常數)

명찰 상수는 문에 전치어로서 쓰여진 표식어이며, 실행 중에, 프로그램 관리(管理)가 명찰을 인용(引用)하여 그 문으로 옮겨질 수 있다. 콜론(colon)은 문에 명찰을 이어준다.

ABC : DIS = RATE * TIME ;

위에서, ABC는 문 명찰(statement label)이다. 이 문은 정상 순서의 실행에 의해서나 GO TO 문을 써서 다른 곳에서 이 문으로 통제를 보내는 것으로 실행될 수 있다. 이 명찰은 두개 이상을 붙일 수 있다.

ABC : EFG : HIJ : A = B ;

〔2〕 명찰 변수(label variable)

문-명찰 변수(statement-label variable)는 문-명찰 상수(statement-label constant)를 인용하는 표식어(標識語 identifier)이다.

보기 :

```
LBLA : 문 ; . . .
LBLB : 문 ; . . .
       LBLX = LBLA ; . . .
       GO TO LBLX ; . . .
       GO TO LBLB ; . . .
```

LBLA와 LBLB는 문에 붙어 있기 때문에 문-명찰 상수이다. LBLX는 문-명찰 변수이다. LBLX에 LBLA를 대입해서, 문 GO TO LBLX는 LBLA 문으로 옮겨가게 한다. LBLX의 값은 이것에 다른 값이 대입될 때까지 그대로 남아 있다.

문-명찰 변수는 LABEL 속성으로 선언되어야 한다.

DECLARE LBLX LABEL ;

2. 지침 자료(指針 資料 pointer data)

지침 자료는 프로그램 통제 자료의 하나이다. 지침 자료 항

목은 지침 변수 (pointer variable)의 값이다. 이는 상수가 없다.

지침 변수는 지침의 이름이며 기초된 기억소 종류의 변수와 연관해서 사용된다. 지침 변수의 값은 기억소안에 있는 자료의 주소라 하겠다.

지침 변수를 선언하는 열쇠말 속성은 POINTER이다. 지침 변수의 사용에 대해서는, 제 I 부, 제 8장 "입력과 출력"과 제 12장 "기초된 변수와 지침변수"를 보라.

제 4 절 자료 구성 (資料 構成 data organization)

PL/I에서, 자료 항목은 한개의 자료 요소 (data element) 가 되거나, 배열 (配列)과 구조체 (構造体 structure) 라 하는 자료 집합체를 이루기 위해서 한데 모을 수 있다. 한개의 요소를 표시하는 변수는 요소 변수 (要素 変数 element variable) (또는 스켈러 변수라고도 함 (scalar variable))이다. 자료 요소의 집합체를 표시하는 변수는 배열 변수 (配列 変数 array variable)이거나 구조체 변수 (構造体 変数 structure variable이다. 어떤 형식의 자료 항목도 -- 산수, 줄, 명찰, 지침 -- 배열이나 구조체로 모아질 수 있다.

1. 배열 (配列 arrays)

〔1〕 배열

같은 특성을 가진 자료 요소들을, 이는 같은 자료 형식과 같은 정도와 길이를 말한다, 배열을 이루기 위해 한데 모을 수 있다. 배열이란 n 차원 (次元 dimension)의 요소

집합체이며, 각 요소는 모두 동일한 속성을 가진다. 배열에만 이름이 주어진다. 배열의 개개의 항목은 배열 안에서 관계적 위치를 부여 하므로써 인용된다.

다음을 생각해 보라:

```
DECLARE  LIST(8) FIXED DECIMAL(3), TABLE(4,2) FIXED
         DECIMAL(3) ;
```

첫번 보기에서, LIST는 8개 요소의 1차원으로 선언되었으며, 각개는 세 자리의 고정점 십진 항목이다. 두번째 보기에서, TABLE은 2차원 배열로 선언되었고, 8개의 고정점 십진 요소들로 되어 있다.

괄호로 싸인 수는 차원 속성(次元 属性 dimension attribute)지정이다. 이는 빈자가 있거나 없이 배열 이름 뒤에 와야 한다.

한개의 차원의 한계(限界 bound)는 그 차원의 끄트머리이다; 한개 차원의 시작은 항상 1이다. 각개의 인용은 첨자로서 나타낸다.

보기:

```
LIST(1), LIST(2), . . .  LIST(8)
```

위에서 LIST 뒤에 붙은 숫자를 첨자(添字 subscript)라 한다.

TABLE은 4행(行 row)과 2열(列 column)의 행렬(行列 matrix)로 설명될 수 있다.

TABLE(m,n)	(m,1)	(m,2)
(1,n)	20	5
(2,n)	10	30
(3,n)	630	150
(4,n)	310	150

자료 항목은 행 대순(行 大順 row major order)으로 배열에 배치된다.

보기 :

TABLE(1,1), TABLE(1,2), TABLE(2,1), TABLE(2,2), . . .

TABLE(4,1), TABLE(4,2) 순서이다.

PL/I D-변성자는 배열에서 최대 3개의 차원을 허용한다. 어떤 배열 요소를 인용함에도, 첨자붙은 이름(subscripted name)은 배열에 있는 차원 수만큼의 첨자를 가져야 한다. 여기서는 산수 자료를 보였으나 여타의 자료 형식도 배열로 된다.

〔2〕 첨자로 쓰는 식(式 expression)

첨자 붙은 이름의 첨자는 상수만이 될 필요는 없다. 유효한 산수 값이 되는 어떤 식도 사용될 수 있다. 만약 그런 식의 값낸 결과가 고른수가 아니라면, 소수 부분은 무시된다. 조직/360에서, 고른수 값은, 만약 필요하다면, 정도(15,0)의 고정점 이진수로 바뀐다. 이는 첨자가 내부에서 이진 고른수로 유지되기 때문이다.

첨자는 자주 변수나 식으로 표현된다. 그러므로, TABLE(I,J*K)가 I, J, K를 바꾸므로써 TABLE의 요소를 인용하는데 사용될 수 있다.

첨자는 식으로 될 수 있으나, 차원 속성 선언의 한계는 부호없는 십진 고른수 상수이어야 한다. 물론 첨자의 값은 한계 이내에 들어야 하며, 안 그러면 오류이다.

2. 구조체(構造体 structures)

〔1〕 구조체

동일한 특성을 갖지 않으나 서로간에 논리적 관계를 원하는 자료 항목은 구조체(structure)라 하는 집합체로 모아질 수 있다. 배열처럼, 전체 구조체는 자료 전체 집합체를 인용하기 위하여 사용되는 이름이 주어진다. 그러나 배열과 같지 않은 점은, 구조체의 각 요소는 또한 이름을 갖는다.

구조체란 이름의 계급적 집합체이다. 최하위 계급에 요소의 집합체가 되고, 그것은 한개의 자료 항목이거나 배열이다. 최상위 계급에 구조체 이름이 되고, 이는 요소의 전체의 집합체를 표시한다.

이름 앞에 숫자를 써서 이름의 계급을 나타낸다. 첫째 수준에 (水準 level) 대구조체 (大構造体 major structure) 가 되고, 좀 낮은 수준에 소구조체 (小構造体 minor structure) 가 되고, 가장 낮은 수준에 요소 (etement) (또는 배열) 가 된다. 대구조체는 수준 번호 1 로서 선언되어야 한다. 소구조체와 요소는 1 보다 큰 수준 번호로서 선언되어야 한다. 이 번호는 십진 고른수 상수이어야 한다. 빈자가 수준 번호와 관련된 이름 사이에 있어야 한다.

보기 :

```
DECLARE 1 PAYROLL, 2 NAME,  3 LAST,     3 FIRST,
                   2 HOURS, 3 REGULAR,  3 OVERTIME,
                   2 RATE,  3 REGULAR,  3 OVERTIME ;
```

위 보기에서, 요소의 속성은 부여되지 않았으나, 태만 대행 (怠慢代行 default assumption) 이 된다. 앞 장 첫머리에서도 말했듯이 모양을 맵씨있게 쓰는 것은 각자의 일이다.

보기 :

DECLARE 1 PAYROLL, 2 NAME,

3 LAST FIXED DECIMAL(6,4), . . .

프로그래머는 이름 PAYROLL로써 전체의 구조체를 인용하거나, 소구조체의 이름으로써 구조체의 일부를 인용할 수도 있다. 요소의 인용이나 배열의 인용은 말할 것도 없다.

보기 :

GET EDIT(PAYROLL, NAME, LAST) (서식-나열) ;

위 보기는 전체와 NAME과 LAST를 인용한 것이다. 고로 LAST는 세번 인용한 것이 된다.

다음으로 낮은 수준을 위해 쓴 번호가 바로 하나 많은 고른수가 되지 않아도 된다. 번호는 단지 이름의 관계적 수준을 지정할 뿐이다. 수준 n의 소구조체는 그 소구조체 이름과 n보다 작거나 같은 수준 번호를 가진 이름 사이에 들어 있는 n보다 큰 수준 번호를 가진 이름을 모두 포함한다. 대구조체 기술(記述)은 수준 번호 1을 가진 다른 항목의 기술이 뒤따르거나, 수준 번호를 갖지 않은 항목의 기술이 뒤따르거나, DECLARE 문을 마치는 세미콜론이 뒤따르는 것으로 끝난다.

보기 :

```
DECLARE 1 PAYROLL,4 NAME FIYED BINARY(20),
          5 LAST, 5 FIRST CHARACTER(15),
          2 HOURS,6 REGULAR, 5 OVERTIME(3),
          2 RATE, 3 REGULAR, 3 OVERTIME,
          COUNTER FLOAT DECIMAL(5),
          1 TA,  2 WA,
          1 TB,  2 WB ;
```

수준 번호는 DECLARE 문에서만 구조체 이름에만 지정된다. 구조체를 인용할 때나 요소를 인용할 때는, 수준 번호가 사용되지 않는다. 구조체만이 수준 번호를 가지고 선언될 수 있다.

〔2〕 수식된 이름(修飾된 이름 qualified names)

소구조체나 구조체 요소는 중복됨이 없다면 그 이름으로써 인용될 수 있다.

그러나 앞에서 이름 REGULAR와 OVERTIME은 구조체 PAYROLL 선언에서 두번 나타났다. 이름의 인용은 그 이름을 일의하게(一意하게)하는 어떤 수식이 없이는 애매할 것이다.

PL/I은 이런 애매성을 피하기 위하여 수식된 이름의 사용을 허용한다. 수식된 이름(修飾된 qualified name)이란 하나 이상의 높은 수준의 이름으로 수식하여 일의하게 된 요소 이름이나 소구조체 이름이다. 앞의 PAYROLL 보기에서, REGULAR와 OVERTIME은 수식된 이름 HOURS•REGULAR, HOURS•OVERTIME, RATE•REGULAR, RATE•OVERTIME을 사용해서 고유하게 된다.

수식된 이름의 다른 이름은 마침표로 이어진다. 빈자가 끼지 못한다. 수식은 수준의 순서이다. 이는 가장 높은 수준에 있는 이름이 먼저 오고, 가장 낮은 수준에 있는 이름이 맨 뒤에 나타난다. 앞에 있는 어떤 이름을 빼도 된다.

대구조체 이름은 고유해야 한다. D-변성자에서, 수식의 첫번째 이름은 틀려야 한다.

보기 :

다음은 유효한 이름이다.

LAST, PAYROLL•NAME•LAST, PAYROLL•LAST, NAME•LAST

보기 :

```
DECLARE 1 PAYROLL,  2 NAME,    3 FIRST,
        3 LAST,     2 HOURS,   3 REGULAR,
        3 OVERTIME, 2 HOURS,   3 REGULAR,
        3 OVERTIME,
```

위와 같을 때, 다음은 오류이다.

```
HOURS.REGULAR
```

이는 고유하지 않기 때문이다.

3. 구조체의 배열(arrays of structure)

구조체의 배열은 D-변성자에서 허용 안 된다. 그러나 구조체의 배열에 유사한 것이 가능하다. PL/I 프로그래머의 안내서 (PL/I programmer's guide)는 이에 대한 기법을 제공한다.

제5절 그 밖의 속성(属性 attributes)

BINARY나 DECIMAL과 같은 자료 변수를 위한 열쇠말 속성은 이장의 앞절에서 간략하게 논술되었다. 여러가지 자료 형식에 공통되는 그 밖의 속성이 적용될 수 있다. 이런 속성에 대한 충분한 논술은 제Ⅱ부, 제9장, "속성"에 있다. 자료 형식과 자료 구성에 대한 논술에 특별히 관계되는 명가지는 ALIGNED, UNALIGNED, DEFINED 이다.

1. ALIGNED와 UNALIGNED 속성

ALIGNED와 UNALIGNED 속성은 기억소에서 자료 요소의 자리잡기를 지정하는 데 사용된다.

만일 ALIGNED 속성이 자료 요소에 지정 되면, 자료 요소는 특

별한 기억소 경계(境界)에 할당된다. 이는 이런 할당이 프로그램 실행에 효과적일때 취한다. 만약 비트-줄 자료가 배열의 꼴로 나타나거나 구조체에 들어 있으면, 이는 명시 선언된 ALIGNED가 있어야 한다.

만약 자료 요소가 UNALIGNED 속성을 받으면, 이것을 앞서는 요소에 잇대서 기억된다.

태만 대행은 요소 수준에 적용된다. 문자 줄 자료와 수치 문자 자료를 위한 태만 대행은 UNALIGNED이며; 그 밖의 모든 형식의 자료를 위해서는, 태만 대행은 ALIGNED 이다.

구조체의 정렬(正列) 안된 요소는 중첩 정의에 유용하다. (다음에 오는 DEFINED 속성을 보라.)

주: D-수준 편성자는 조직/360 에서 자(字 byte) 정렬을 허용하는 자료에는(이는 문자 종류 자료) UNALIGNED 만을 제공하므로, UNALIGND 또는 ALIGNED의 지정은 어떤 경우에는 틀리거나 효과가 없다. D-수준 편성자는 비트 줄을 위한 태만 대행 UNALIGNED를 해 주지 않고, ALIGNED로 고정시켰다. F-수준 편성자는 UNALIGNED를 쓴다.

2. DEFINED 속성

DEFINED 속성은 명명된 자료 요소, 구조체, 배열이 두개 자료가 배치된 동일 기억역(記憶域)이 인용됨을 지정한다.

보기 :

```
DECLARE LIST(100,100), LISTA(100,100) DEFINED LIST ;
```

위 결과는 LISTA에 있는 요소를 인용함은 LIST에 있는 대응하는 요소를 인용함과 같다. 이런 형식의 정의(定義)를 대응

정의(対応 定義 correpondence defining)이라 한다.

다른 형식의 정의를 중첩 정의(重畳 定義 overlay defining)라 한다. 이런 형식의 정의는 정의된 항목(定義된 項目 defined item)(DEFINED 속성을 가진 항목;보기로 LISTA)이 기초 표식어(基礎 標識語 base identifier)(열쇠말 DEFINED 다음에 오는 표식어;보기로 LIST)에 의해 차지된 기억소의 전부 또는 일부를 인용할 것임을 지정한다.

보기:

```
DECLARE 1 P, 2Q CHARACTER(25),
          2R CHARACTER(50),
        STR CHARACTER(60) DEFINED P ;
```

이 보기에서, STR은 구조체 P에서 정의된 길이 60의 문자 줄이다. P의 요소는 정렬이 안되었으므로(unaligned), Q에 있는 첫번째 부터 R에 있는 마지막까지의 모든 문자가 75 문자 길이의 한개의 줄로 여겨진다. STR은 이줄의 처음부터 60개 문자를 인용한다. 이것은 Q의 25 문자에 R의 35 문자를 이은 것이다. 만일 P의 요소가 정렬 안된 것(unaligned)이 아니면, STR의 내용은 장담할 수 없다.

3. INITIAL 속성

INITIAL 속성은 기억소가 이것에 할당될 때 변수에 대입되는 초기 상수 값을 지정한다.

보기:

```
DECLARE NAME CHARACTER(10) INITIAL('JOHN DOE'), PI
FIXED DECIMAL(5.4) INITIAL(3.1416) ;
```

기억소가 NAME에 배당될 때, 문자 줄 'JOHN DOE' (10으로 채워진다.)가 이것에 대입된다. PI가 배당될 때, 이는 값 3.1416으로 시작된다.

프로그램이 실릴 때 배당되는 변수에 대해서는, 이런 변수는 STATIC 변수이며 프로그램의 실행동안 기억소에 그대로 남아 있다. INITIAL 속성에서 지정된 값은 한번 대입된다. AUTOMATIC 변수에 대해서는, 이는 선언한 블러크의 기동시마다 배당된다. 지정된 초기치가 매(每) 배당시(配当時)에 대입된다.

INITIAL 속성은 배열에도 지정될 수 있다. 구조체 선언에서, 요소 이름만이 INITIAL 속성을 받을 수 있다. 배열은 부분에만 초기치를 받거나 전부가 받거나 할 수 있다.

보기 :

```
DECLARE A(15) CHARACTER(13) INITIAL('JOHN DOE', 'RICHA-
RD ROW', 'MARY SMITH'),
        B(10,10) DECIMAL FIXED(5) INITIAL((25)0, (25)1,
(50)0) ;
```

A는 처음 세 요소만이 초기치를 갖는다. 배열 B는 전부 초기치를 갖는다. 처음 25개는 0으로, 다음 25개는 1로, 마지막 50개는 0으로 시작된다. 괄호로 싸인 수(25, 25, 50)은 중복 인수(重複 因数 iteration factor)이며 ; 이들은 초기치(初期値)를 받는 요소의 수를 지정한다.

중복 인수는 1과 같거나 큰 십진 고른수 상수이어야 한다. 이는 앞 절에서 논술된 줄 반복 인수(줄 反複 因数 string repetition factor)와 혼동해서는 안된다.

보기 :

```
DECLARE TABLE(50) CHARACTER(10) INITIAL((10)'A', (25)
```

(10)'B', (24)(1)'C') ;

이 INITIAL 속성 지정은 중복 인수와 반복 인수를 다 가지고 있다. 첫번째 요소는 10개의 A가 들어가고, 다음 25개에는 10개의 B가 들어가고, 마지막 24개에는 1개의 C가 들어간다.

INITIAL 속성 지정에서, 줄 배열에 대해서는, 줄 상수 앞에 오는 한개의 인수는 줄 반복 인수(repetition factor)로 간주된다 ((10)'A' 에서 처럼). 두개가 나타나면, 첫째 것은 중복 인수(重複 因数 iteration factor)로 간주되고, 두번째 것은 줄 반복 인수로 간주된다.

제 4 장 식 (式 expression)

식이란 값의 표현이다. 하나의 상수나 변수는 하나의 식이다. 연산자 (演算子 operator) 와/또는 괄호로 이어진 상수와 /또는 변수의 결합은 식이다. 연산자를 포함한 식은 연산 식 (演算 式 operational expression) 이다. 연산 식의 상수와 변수를 연산항 (演算項 operand) 이라 한다.

보 기 :

27, LOSS, A + B , (SQTY-QTY) * SPRICE

어떤 식도 요소 식 (要素 式 element expression) (또는 스칼라 식 (scalar expression) 이라고도 함) , 배열 식 (配列 式 array expression) , 또는 구조체 식 (構造体 式 structure expression) 으로 분류된다.

요소 식은 요소 값을 나타낸다. 배열 식은 배열값을 표현하는 것이다.

구조체 식은 구조체 값을 표현한 것이다.

배열 변수와 구조체 변수는 같은 식에 나타나지 못한다. 그러나, 요소 변수와 상수는 배열 식이나 구조체 식 어느쪽에도 나타날 수 있다. 구조체 안에 있는 요소 이름이나 배열중의 한개 요소를 지정하는 첨자붙은 이름은 요소 식이다.

보 기 :

```
DECLARE A(10,10) BINARY FIXED(31), B(10,10)
BINARY FIXED(31),
1 RATE, 2 PRIMARY DECIMAL FIXED(4,2), 2 SECONDARY
DECIMAL FIXED (4,2),
```

1 COST, 2 PRIMARY DECIMAL FIXED(4,2), 2 SECONDARY DECIMAL FIXED (4,2),
C BINARY FIXED(15), D BINARY FIXED(15):

요소 식의 보기 :

C*D, A(3,2)+B(4,8), RATE·PRIMARY-COST·PRIMARY
A(4,4) *C RATE·SECONDARY/4, A(4,6) * COST·SECONDARY

배열 식의 보기 :

A+B, A*C-D, B/IOB

구조체 식의 보기 :
RATE * COST, RATE/2

제1절 식의 이용(利用)

하나의 상수나 하나의 변수인 식은 프로그램 어디서나 자유로이 나타날 수 있다. 그러나, 많은 PL/I 문의 구문(構文 syntax)에는 연산 식의 출현을 허용하며, 이 식은 식의 계산값이 유효한 값이 되어야 한다.

이 책에서 사용된 구문에 대한 설명에서, 별명이 없는한 용어 "식"은 요소 식, 배열 식, 구조체 식을 말한다. 식의 종류가 제한된 경우에는, 제한의 형식이 강조될 것이다. 보기로, 용어 "요소-식"은 구문 설명에서 배열 식이나 구조체 식은 무효하다는 것을 가르킨다.

주 : 이 식은 대입문에서의 사용이 매우 많다.
A = B + C ;

제2절 연산 식에서 자료 변환 (資料 変換 data Conversion in OPerational expresions)

연산 식은 하나 이상의 단일 (単一) 연산으로 이루어진다. 단일 연산이란 하나의 전치 연산 (前置 演算 prefix operation) (한개의 연산항의 앞에 오는 연산자) 이나 하나의 삽입 연산 (挿入 演算 infix operation) (두 연산항의 가운데 있는 연산자) 이다. 연산이 이루어질 때 , 어떤 삽입 연산의 두개의 연산항도 하나의 변수나 상수의 속성에 의해 지정된 동일한 자료 형식으로 되어야 한다.

PL/I 식에 있는 연산의 연산항은 , 필요하면 , 연산이 이루어지기 전에 자동으로 공통된 표현으로 바뀐다. 여러가지 자료 형식의 변환을 위한 일반적 규칙은 이 다음의 문장과 다음 절에 있는 〃자료 변환의 개념〃에서 논술된다. 특정한 경우의 상세한 규칙은 , 변환된 항목의 정도 (精度) 와 길이를 계산하는 규칙을 포함해서 , 제Ⅱ부 , 제6장 , 〃자료 변환〃에서 찾아볼 수 있다.

자료 변환은 문제 자료 (Problem data) 의 변환에 국한한다. 문명찰과 지침과 같은 프로그램 통제 자료는 한가지 형식에서 다른 형식으로 바뀌지 않는다.

〔1〕 비트-줄을 문자-줄로 (bit-string to character-string)

비트 1은 문자 1이 되고; 비트 0은 문자 0이 된다.

〔2〕 문자-줄을 비트-줄로 (character-string to bit-string)

문자 줄은 문자 1과 0만으로 되어 있어야 한다. 이런

경우 문자 1은 비트 1이 되고, 문자 0은 비트 0이 된다. CONVERSION 조건(条件 Condition) 이 0과 1이 아닌 문자를 비트로 바꾸려고 하면 일어난다.

〔3〕 문자-줄을 산수로(Character-string to arithmatic)

문자-줄 자료는 규약 산수나 수치 문자 형식으로 바뀌지 않는다. 이런 시도는 오류이다.

〔4〕 산수를 문자-줄로 (arithmatic to character string)

규약 산수 자료는 문자-줄 형식으로 바뀌지 않는다. 이런 시도는 오류이다. 그러나, 수치 문자 자료는 문자 줄로 바뀔 수 있다. 수치 문자난은 같은 문자를 가진 문자 줄로 옮겨진다. 줄의 길이는 수치 문자 난을 위한 PICTURE 속성에 지정된 길이와 같다.

〔5〕 비트-줄을 규약 산수로 (bit-string to coded arithmatic)

비트 줄은 부호없는 이진 고른수로 해석되고 양의 (陽의) 고정점 이진수로 바뀐다. 필요하면, 기수와 척도(基数와 尺度 base and scale)가 또 바뀐다.

〔6〕 비트 줄을 수치 문자로(bit string to numeric character)

비트 줄은 먼저 규약 산수로 바뀌고 다음에 수치 문자로

〔7〕 규약 산수를 비트-줄로(coded arithmatic to bit - string)

필요하면, 절대 값이 고정점 이진수 고른수로 바뀐다. 부호를 무시하고, 이 고른수는 비트 줄로서 바뀐다. 비트 줄의 길이는 처음의 바뀌지 않은 산수 자료 항목의 정도에 좌우된다.

〔8〕 수치 문자를 비트 줄로 (numeric character to bit string)

수치 문자 값이 규약 산수로 바뀌고 다음에 비트 줄로 바뀐다.

〔9〕 수치 문자를 문자 줄로 (numeric to character string)

위에 있는 〃산수를 문자 줄로〃를 보라.

〔10〕 산수의 기수와 척도 변환 (arithmatic base and scale conversion)

산수의 기수 (基数 base) 나 척도 (尺度 scale) 변환의 결과의 정도는 처음의 산수 자료 항목의 정도에 좌우된다. 이 규칙은 제Ⅱ부, 제6장, 〃자료 변환〃에 나열되어 있다.

〔11〕 대입에 의한 변환

식의 값을 내는 연산의 결과로서 이루어지는 변환에 추가해서, 변환이 자료 항목이 -- 또는 식의 값낸 결과 -- 이 항목의 속성과 틀리는 속성을 가진 변수에 대입될 때 일어난다. 이런 변환에 대한 규칙은 일반으로 위에서 논술된 그것과 제Ⅱ부, 제6장, 〃자료 변환〃에 있는 것과 같다.

제3절 식의 연산 (式의 演算 expression operations)

하나의 연산 식은 하나 이상의 단일 연산을 지정할 수 있다. 연산의 종류는 이 연산에 지정된 연산자의 종류에 따른다. 네 종류의 연산이 있다. ……

산수 (算数 arithmatic), 비트-줄, 비교 (比較 compaicson), 잇기 (coneatemation).

1．산수 연산 (arithmatic operation)

산수 연산은 다음 연산자 중의 하나와 결합된 연산항 (演算項 operand) 으로 지정된 것이다.

＋，－，＊，／，＊＊

더하기 기호와 빼기 기호는 전치 연산자 (前置 演算子 Prefix operator) (하나의 연산항과 연관되고 그 앞에 오는 것，＋A 나 －A와 같이) 나 삽입 연산자 (揷入 演算子 infix operator) (두 연산항과 연관되고 그 사이에 오는 것，A＋B나 A－B와 같이) 로서 나타날 수 있다. 다른 산수 연산자들은 삽입 연산자들로서만 나타날 수 있다.

더 복잡한 식은 그 같은 산수 연산의 집합으로 된다. 전치 연산자는 삽입 연산의 어떤 연산항의 앞에 와서 연관될 수 있음을 유의하라. 보기로，식 A＊－B에서，변수 B앞에 오는 빼기 기호는 A의 값이 B의 음의 값으로 곱해지는 것을 가르킨다.

두개 이상의 전치 연산자가 하나의 변수 앞에 와서 연관될 수 있다. 두개 이상의 양의 (陽의) 전치 연산자는 누적 효과가 없을 것이나，두개의 연속한 음 (陰) 전치 연산자는 한개의 양 전치 연산자와 같은 효과를 가질 것이다.

보 기：

－A，－－A，－－－A

〔1〕 산수 연산에서 자료 변환

산수 연산의 두개 연산항은 형식，기수，척도，정도에서 다를지 모른다. 그들이 다를 때，변환은 아래에 열거된 규칙에 따라서 일어난다. 어떤 다른 규칙은－－다음에 나오듯이－－지수식의

경우에 적용된다.

형식(形式 type) : 수치 문자 난 연산항(문자 꼴로 기록된 숫자)과 비트 줄 연산항은 내부 규약 산수 형식으로 바뀐다. 산수 연산의 결과는 언제나 규약 산수 꼴(規約 算数 꼴 coded arithmatic form)로 된다. 형식 변환은 산수 전치 연산에서만 일어날 수 있는 변환이라는 것을 유의하라.

기수(基数 base) : 만일 두 연산항의 기수가 틀리면, 십진수 연산항은 이진수로 바뀐다.

정도(精度 precision) : 만일 정도만이 다르면, 형식 변환이 필요없다.

척도(尺度 scale) : 만일 두 연산항의 척도가 틀리면, 고정점수 연산항은 부동점수 척도로 바뀐다. 이 규칙에 대한 예외는 첫째 연산항이 부동점수 척도이고 둘째 연산항(연산의 지수(指数 exponent)이 영의 척도 인자(scale factor)를 가진 고정점수 {이는 정도(P,0)인 고정점 고른수 상수이거나 변수를 말함, 일 때의 지수식의 경우이다.

만일 지수식 연산의 두개 연산항이 모두 고정점수이면, 변환은 다음과 같이 일어난다.

① 만약 지수(exponent)가 (P,∅) 이외의 정도를 가지면 두 연산항은 부동점수로 바뀐다.

② 지수가 부호없는 고정점 고른수 상수가 아니면 첫째 연산항이 부동점수로 바뀐다.

③ 만일 정도가 고정점수 지수식의 결과가 허용된 최대 수(조직/360에서, 15개 십진수 자리 또는 31개 이진수 자리)를 넘을 것임을 가르키면 첫째 연산항은 부동점수로

바뀐다. 지수식 (exponentiation) 변환에 대한 더 상세한 것과 보기는 이 장에 있는 〃자료 변환의 개념〃 절에 있다.

〔2〕 산수 연산의 결과

앞으로 사용되는, 산수 연산의 〃결과〃란 만약 연산이 한 개의 연산 식에서 지정된 여러개 연산중에 하나이면 하나의 중간 결과라고 보아도 된다. 어떤 결과도 만약 이것이 다음에 오는 연산의 연산항으로 사용되는 중간 결과이거나 이것이 변수에 대입될 것이라면 더 이상의 변환을 요할지도 모른다.

요구되는 변환이 일어난 뒤에, 산수 연산이 이루어진다. 만약 최대 정도를 넘어서 자르기가 필요하면, 자르기는 연산항의 척도와 기수에 상관없이 저위 (底位) 소수 자리에서 이루어진다.

그러나, 고정점수 자료를 포함하는 어떤 경우에서는, 고위 (高位) 숫자가 척도 인자 (scale factor) 가 소수점 배당이 선언된 숫자 자리수에 용납되지 않는 때면 떨어져나갈 수 있다.

결과의 기수, 척도, 정도는 참가한 연산항과 연산자에 의존한다.

전치 연산에서, 결과는 동일한 기수, 척도, 정도를 변환된 연산항처럼 갖는다. A가 비트 줄일 때, -A의 결과는 산수 결과이다.

따라서 A는 연산이 이루어질 수 있기 전에 먼저 규약 산수꼴로 바뀌어야 한다.

삽입 연산자에서, 결과는 다음과 같이 연산항의 척도에 의존한다.

부동 소수점 (floating point) : 만약 삽입 연산의 변환되는 연산항이 부동점수 척도이면, 결과는 부동점수 척도이고, 결과의 기수는 연산항의 공통 기수이다.

결과의 정도는 두 연산항 중에 큰 것이 된다.

고정 소수점 (fixed point) : 만약 삽입 연산의 변환되는 항목이 고정점수 척도이면, 결과는 고정점수 척도이고, 결과의 기수는 연산항의 공통 기수이다. 고정점수 결과의 정도는 아래 열거된 규칙에 따라 연산항에 의존한다.

정도를 산출하는 공식에서, 사용된 기호는 다음과 같다 :

P 결과의 전체 자리수를 표시

q 결과의 척도 인자를 표시

P_1 첫째 연산항의 전체 자리수를 표시

q_1 첫째 연산항의 척도 인자를 표시

P_2 둘째 연산항의 전체 자리수를 표시

q_2 둘째 연산항의 척도 인자를 표시

더하기와 빼기 (addition and subtraction) : 결과에서 전체 자리수는 1 더하기 큰 고른수 자리수 더하기 큰 소수 자리수와 같다. 전체의 자리수는 허용된 최대 자리수를 넘지 못한다 (15개 십진수 자리수, 31개 이진수 자리수), 척도 인자는 두 연산항 중에 큰 쪽의 척도 인자와 같다.

공식 (公式 formula) :

$P = 1 +$ 최대 $(P_1 - q_1 , P_2 - q_2) +$ 최대 (q_1 , q_2)

$q =$ 최대 (q_1 , q_2)

보 기 :

12354.2385 + 222.11111

A B C D

P = 1 + 최대 (5 , 3) + 최대 (4 , 5)

= 1 + 5 + 5 = 11

q = 최대 (4 , 5)

= 5

그러므로 정도는 P (11 , 5) 이다.

곱하기 (multiplication) : 결과에서 전체의 자리수는 1 더하기 첫째 연산항 자리수 더하기 둘째 연산항 자리수와 같다.

공 식 :

$P = 1 + P_1 + P_2$

$q = q_1 + q_2$

보 기 :

345·432 * 22·45

A B C D

P = 1 + 6 + 4

= 11

q = 3 + 2

5

그러므로 정도는 P (11 , 5) 이다.

나누기 (division) : 몫 (quotient) 의 전체 자리수는 허용된 최대수 (15 의 십진 , 31 의 이진) 와 같다. 몫의 척도 인자 (소수 자리수) 는 나누어지는 수 (dividend) 의 고른수 자리수와 나눗수 (divisor) 의 소수 자리수에 좌우된다. 소수 자리수는 전체 자리수에서 나누어지는 수의 고른수 자리수와 나누는 수 (나눗수) 의 소수 자리수를 뺀 것이다.

공 식 :

P = 15 자리의 십진수 , 31 자리의 이진수

$q = 15$ (또는 31) $-$ (($P_1 + q_1$) $+ q_2$)

보 기 :

432·432/2

A B C D

P = 15 (십진수이므로)

q = 15 - 3 - 0 = 12

∴ 정도는 P (15 , 12) 이다.

지수식 (指数式 exponentiation) : 만일 둘째 연산항 (지수) 이 정도 (P , 0) 의 부호없는 영이 아닌 고정점 상수이면, 결과의 전체 자리수는 첫째 연산항 (밑 底 base) 에 1을 더해서 이를 지수 값으로 곱한 것에서 1을 뺀 것이다. 결과의 소수 자리수는 첫째 연산항의 자리수 곱하기 지수 값이다.

주 : 지수식 연산 X**y 에서, 어떤 경우는 다음과 같다.

① X = 0 이고, y > 0 이면, 결과는 0 이다.

② X = 0 이고 y ≤ 0 이면, ERROR 조건이 일어난다.

③ X ≠ 0 이고 y = 0 이면, 결과는 1 이다.

④ X < 0 이고 y가 정도 (P , 0) 인 고정점수이면, ERROR 조건이 일어난다.

("산수 연산에서 자료 변환" 에서 지적했듯이, 만일 지수가 부호없는 고정점수 고른수가 아니거나, 결과의 전체 자리수가 15자리의 십진수 또는 31자리의 이진수를 넘게 되면, 첫째 연산항은 부동점수로 바뀌고 부동점수 지수식 규칙이 적용된다.)

공 식 :

$P = ((P_1 + 1) * (지수 \ \ 값)) - 1$

$q = q_1 * (지수 \ \ 값)$

보 기 :

32 * * 5

$P = ((2 + 1) * 5) - 1 = 14$

$q = 0 * 5 = 0$

∴ 정도 P (14, 0) 이다.

2 . 비트-줄 연산 (bit-string operations)

비트-줄 연산은 다음 연산자 중에 하나와 결합된 연산항으로 지정된 것이다 :

¬ , ε (&) , |

첫째 연산자 〃아니다〃 부호 (〃 not 〃 symbol) 는 전치 연산자로만 쓰인다. 〃그리고〃 부호 (〃 and 〃 symbol) 와 〃또는〃 부호 (〃 or 〃 symbol) 는 삽입 연산자로만 쓰인다. 연산자들은 불 대수 (Bool 代數 Boolean algebra) 에서의 기능과 같다.

비트-줄 연산의 연산항은 필요할 때는 연산이 이루어지기 전에 비트 줄로 바뀐다. 삽입 연산의 연산항이 틀리는 길이면, 짧은 것이 오른 편이 0으로 채워져 늘어난다. 비트-줄 연산의 결과는 길이에서 두 연산항의 길이와 같다. 비트-줄 연산은 한개 비트 단위로 이루어진다.

〃아니다〃 연산자의 효과는 비트를 바꾸는 것이다 ; 이는 ¬1 의 결과는 0이고, ¬0의 결과는 1이다.

"그리고" 연산의 결과는 양쪽이 1이면 1이고, 그 밖에는 0이다. "또는" 연산의 결과는 적어도 한쪽이 1이면 1이다.

다음 표는 이해를 돕겠다.

A	B	¬A	¬B	A ⊄ B	A ∣ B
1	1	0	0	1	1
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	0	0

하나 이상의 비트-줄 연산이 한개의 식에 결합될 수 있다.

보 기 :

원시 자료가 다음과 같다.

A '010111'B

B '111111'B

C '110'B

¬A '101000'B

¬C '001'B

C ⊄ B '110000'B

A ∣ B '111111'B

C ∣ B '111111'B

A ∣ ¬C '011111'B

¬(¬C/¬B) '110111'B

3 . 비교 연산 (比較 演算 comparison operations)

비교 연산은 다음 연산자 중의 하나와 결합된 연산항으로 지정된다.

< , ¬< , <= , = , ¬= , >= , > , ¬>

이 연산자는 차례로 〃보다 작다〃 (〃 less than 〃) , 〃보다 안 작다〃 (〃 not less than 〃) , 〃보다 작거나 같다〃 (〃 less than or equal to 〃) , 〃같다〃 (〃 equal to 〃) , 〃안같다〃 (〃 not equal to 〃) , 〃보다 크거나 같다〃 (〃 greater than or equal to 〃) , 〃보다 크다〃 (〃 greater than 〃) , 〃보다 안크다〃 (〃 not greater than 〃) 이다.

비교에는 네가지 종류가 있다.

① 대수 (代數 algebraic)

이는 내부 규약 산수꼴로 있는 부호있는 산수값의 비교이다. 만일 기수 , 척도 , 정도가 다르면 , 산수 연산의 규칙에 따라 바뀐다. 수치 문자 자료는 비교되기 전에 규약 산수로 바뀐다.

② 문자

이는 왼쪽에서 오른쪽으로 , 대조 순위에 따라 한 문자씩 문자의 비교이다.

③ 비트 (bit)

이는 외쪽에서 오른쪽으로 , 한 비트씩의 비교이다.

④ 지침 (指針 pointer)

이는 =와 ¬=만이 허용된다. 양쪽의 연산항은 유효한 지침 식 (pointer expression) 이어야 한다. 그러므로

프로그램 통제자료의 형식 변환은 없다.

만일 비교의 연산항이 (지침은 제외) 같지 않은 형식이면, 낮은 형식의 연산항이 높은 형식의 연산항의 형식으로 바뀐다. 형식의 우선 순위는 〔1〕 내부 규약 산수(제일 높음), 〔2〕 문자 줄, 〔3〕 비트 줄이다. (문자 줄은 산수 자료와 비교되지 못한다)

만일 문자-줄 연산의 연산항이 변환뒤에 길이가 틀리면, 짧은 연산항이 오른쪽이 빈자로서 늘어난다. 비트인 경우는 0으로 늘어난다.

비교 연산의 결과는 언제나 길이가 하나인 비트 줄이다; 이 값은 만일 관계가 사실이면 '1'B이고, 사실이 아니면 '0'B이다.

비교 연산의 대부분은 IF문에 나타난다.

```
IF  A=B
        THEN 사실이면 취할 행위
        ELSE 허위이면 취할 행위
```

식 A=B의 값은 '1'B이나 '0'B이 된다. 이 값에 따라 IF문의 THEN부분 또는 ELSE 부분이 실행된다.

그러나 비교 연산은 IF 문에 한하지 않는다. 다음 대입문은 유효하다;

```
X = A<B ;
```

이 보기에서, 만일 A가 B보다 적으면 X에 '1'B가 대입되고, 아니면 '0'B가 대입된다. 같은 방법으로 다음도 유효하다.

```
X = A=B ;
```

4 . 잇기 연산 (concatenation operations)

잇기 연산은 잇기 부호와 결합된 연산항으로 지정된다 :

||

이는 왼쪽 연산항의 마지막 문자나 비트 다음에 오른쪽 연산항의 첫째 문자나 비트를 이어서 두 연산항이 결합됨을 의미한다.

잇기는 줄 (문자 줄 또는 비트 줄) 또는 PICTURE 변수에만 이루어진다.

만일 양쪽이 문자 줄이거나 비트 줄이면 , 변환은 일어나지 않는다. 그렇지 않으면 양쪽 연산항이 문자 줄로 바뀐다.

잇기 연산의 결과는 다음과 같다.

비트 줄 (bit string) : 두 비트-줄 연산항 길이의 합계 길이인 비트 줄.

문자 줄 : 두 문자-줄 연산항 길이인 문자 줄.

보 기 :

원시 자료가 다음과 같다.

A '010111 'B , B '101 'B ,

C 'Xy,Z', D 'AA/BB'

A || B는 '010 111101 'B

A || A || B는 '010 111 010 111 101 'B

C || D 는 'Xy,ZAA/BB'

D || C 는 'AA/BB Xy,Z'

B || D 는 '101AA/BB'

마지막 보기에서 , 비트 줄 '101 'B는 문자 줄 '101'로 바뀌었다. 결과는 문자 줄이다.

5 . 연산의 결합

서로 다른 연산의 형식이 하나의 연산 식에서 결합될 수 있다. 어떤 결합도 사용될 수 있다.

보 기 :

```
RESUL T = A + B < C & D ;
```

식에 있는 각 연산은 당해 종류의 연산 규칙에 따라 값이 내지며, 연산이 이루어지기 앞서 필요한 자료 변환이 된다.

위에 있는 변수를 다음과 같이 정한다.

```
DECLARE   RESULT  BIT(3),  A  FIXED  DECIMAL (1),
          B  FIXED  BINARY (3),   C BIT (2),
          D  BIT (4);
```

위 식은 다음과 같이 이루어진다.

- O A의 십진수 값은 이진수로 바뀐다.
- O 이진수 더하기가 이루어진다. A와 B를 더함으로써
- O 이진수 결과가 C의 변환된 이진수와 비교된다.
- O 비교의 비트-줄 결과가 비트 줄 D의 길이로 늘어나서, "그리고" 연산이 이루어진다.
- O "그리고" 연산의 결과가, 길이 4의 비트 줄이다. 변환없이 RESULT에 대입된다.

주 : 한개 식 안의 값내기 순서는 식에 나타난 연산자의 우선순위에 따른다.

- O 위 보기에서, 연산의 우선순위는 왼쪽부터 오른쪽 순이다.

연산자의 우선순위 (憂先順位 priarity of operaters)

식의 값내기에서 , 연산자의 우선순위는 다음과 같다.

① ＊＊ 전치＋ , 전치－ , ㄱ (최상위)

② ＊ , ／

③ 삽입＋ , 삽입－

④ ||

⑤ ＜ , ㄱ＜ , ＜＝ , ＝ , ㄱ＝ , ＞＝ , ＞ , ㄱ＞

⑥ ∈

⑦ 1 (최하위)

번호간의 순위는 1 , 2 , 3 , 4 , 5 , 6 , 7 이다. 같은 번호 안에서 , 1 번은 오른쪽이 왼쪽보다 우선하고 , 나머지 번호 안에서는 왼쪽이 오른쪽보다 우선한다.

여기서 덧붙일 것은 괄호는 안에 들은 것을 먼저 처리한다.

보 기 : 위 문제를 보기로 들겠다.

```
RESULT = A + B < C ∈ D ;
```

이는 다음 순이다.

```
( A ) + ( B )
( A + B ) < ( C )
( A + B < C ) ∈ ( D )
```

이를 다음과 같이 괄호를 쓰면 순위는 달라진다.

```
RESULT = ( A + B ) < ( C ∈ D ) :
```

몇개의 보기를 설명없이 들겠다.

```
A + ( B < C ) ∈ ( D | E * F ) , A + B * * 3 ∈ C * D - E ,
A + B * * ) ∈ ( C * D - E ) , IF ( A = B ) OR C = D
THEN ;
```

제 4 절 배열 식 (配列 式 anay expressions)

배열 식이란 한개의 배열 변수이거나 적어도 한개의 배열 연산항을 포함하는 식이다. 배열 식은 또한 연산자와 요소 변수, 상수들을 포함할 수 있다.

배열식의 값낸 것은 배열 결과가 된다. 배열에 이루어지는 모든 연산은 행대순 (行大順) 으로 하나의 요소씩 이루어진다. 고로 배열식에 쓰인 모든 배열은 같은 한계를 가져야 한다.

주 : 더하기와 빼기를 뺀 행렬식은 약정된 행렬 (matrix) 대수식이 아니다.

1. 전치 연산자와 배열 (前置 演算子와 配列 prefix operators and arrays)

배열에서 전치 연산자가 있는 연산의 결과는 동일한 한계 (bound) 의 배열이며, 각 요소는 처음 배열의 각 요소에 이루어진 연산의 결과이다.

보 기 :

$$A \quad \begin{matrix} 5 & 3 & -9 \\ 1 & -2 & 7 \\ 6 & 3 & -4 \end{matrix}$$

$$-A \quad \begin{matrix} -5 & -3 & 9 \\ -1 & 2 & -7 \\ -6 & -3 & 4 \end{matrix}$$

2．삽입 연산자와 배열（挿入 infix）

이루어진 연산의 결과이다.

보 기：

```
A       5   10   8
       12   11   3

A＊3은   15   30   24
         36   33    9
```

배열 변수를 포함한 삽입 연산은 한쪽 연산항으로서 요소나 다른 배열을 가져도 된다.

〔1〕 배열과 요소 연산

요소와 배열이 삽입 연산자로 결합된 연산의 결과는 처음 배열과 동일한 한계를 가진 배열이다. 각 요소는 처음 배열의 대응하는 요소와 단일 요소와

A＝A＊A（1，2）；는

```
A       50     100     800
      1200    1100     300
```

〔2〕 배열과 배열 연산

만일 두배열이 삽입 연산자로 이어지면，두 배열은 같은 수의 차원과 한계를 가져야 한다. 결과는 처음 배열의 그것과 같은 한계를 가진 배열이다. 연산은 두 배열의 대응하는 요소에 이루어진다.

보 기：

```
A       2    4    3
        6    1    7
        4    8    2
```

B 1 5 7

8 3 4

6 3 1

A＋B는 3 9 10

14 4 11

10 11 3

A＊B는 2 20 21

48 3 28

24 24 2

〔3〕 배열 식에서 자료 변환

결합된 연산과 자료변환의 규칙은 요소 연산의 그것과 같다.

제5절 구조체 식(構造体 式 structure expressions)

구조체 식이란 단일의 구조체 변수이거나 적어도 하나의 구조체 연산항을 포함하며 배열 연산항이 들어있지 않은 식이다. 요소 변수나 상수가 구조체 식의 연산항이 될 수 있다. 구조체 식의 값낸 결과는 구조체가 된다. 구조체 연산항은 대구조체 이름 또는 소구조체 이름이 될 수 있다.

구조체에 이루어지는 모든 연산은 한개 요소씩 이루어진다. 구조체 식의 모든 구조체 연산항은 동일 구조를 가져야 한다.

동일 구조란 구조체가 동일한 소구조와 같은 수의 요소와 배열을 가져야 하며 구조체 안에서 요소와 배열의 자리잡은 것이 같

아야 한다. 배열은 동일 한계를 가져야 한다. 이름이 같을 필요는 없다. 대응하는 요소의 자료 형식이 유효한 변환이 이루어질 수 있다면, 같을 필요는 없다.

1. 전치 연산자와 구조체

구조체에 대한 전치 연산자의 연산 결과는 동일 구조를 가지는 구조체이며, 각 요소는 처음 구조체의 각 요소에 이루어진 연산의 결과이다.

주 : 구조체는 여러가지 틀리는 형식의 자료를 가질 수 있으므로 구조체 식에 있는 전치 연산은 이 연산이 유효하게 이루어지기 전에는 의미가 없다.

2. 삽입 연산자와 구조체

한개의 연산항으로서 구조체 변수를 포함하는 삽입 연산은 요소나 다른 구조체를 다른쪽 연산항으로서 가질 수 있다.

〔1〕 구조체와 요소 연산

연산이 한개의 구조체와 한개의 요소 연산항을 가지면, 구조체에 있는 각 요소와 이 요소간의 연속한 연산과 같다.

보 기 :

```
1 A, 2 B, 3 C, 3 D, 3 E, 2 F, 3 G,
  3 H, 3 I

A*X는

C*X, D*X, E*X, H*X I*X

소구조체를 쓸 수도 있다.

B ε '1010'B, F*32
```

〔2〕 구조체와 구조체 연산

연산이 두 구조체 연산항을 가지면, 이는 대응하는 요소의 짝에 대한 요소 연산의 연속과 같다.

보 기 :

1 M , 2 N , 3 O , 3 P , 3 Q , 2 R , 3 S

3 T , 3 U ,

A ‖ M 이면

C ‖ O , D ‖ P , E ‖ Q , G ‖ S , H ‖ T , I ‖ U

다음도 구조체 식이다.

N ∈ R

제6절 식의 연산항 (式의 演算項 operands of expression)

식의 연산항은 상수, 요소 변수, 배열 변수, 또는 구조체 변수가 될 수 있다. 연산항은 또 계산 결과이며 하나의 값을 표현하는 식이 될 수 있다.

보 기 :

A = B * SQRT (C) ;

이 보기에서, 식 SQRT (C) 는 C값의 제곱근과 같은 값을 나타낸다. 이런 식을 함수 인용 (函数 引用 function reference) 이라 부른다.

1. 함수 인용 연산항 (函数 引用 演算項 function refence operands)

함수 인용은 이름과 괄호에 싸인 하나 이상의 상수, 변수, 또는 다른 식의 나열로 구성된다. 이름은 나열 표현된 자료에 지

정된 계산이 이루어지고 함수를 인용한 자리에서 계산된 값으로 대치되도록 씌어진 약정된 블러크의 이름이다.

위 보기에서, C가 16이라면, 함수 인용 SQRT(C)는 16의 제곱근을 계산해서 함수 인용을 값 4로 바꿔놓는 실행이 된다.

함수 인용에서 이름으로 표현되는 부분을 함수(函数 function)라 부른다. 함수 SQRT는 내조립 함수(内組立 函数 built-in function)의 하나이다. 여러가지 연산을 제공하는 내조립 함수는 PL/I 언어의 일부분이다. 자세한 논술은 제Ⅱ부, 제7장, "내조립 함수와 유사 변수"에 나타난다. 덧붙여서, 프로그래머는 다른 목적으로 쓸 수 있고 (제10장, "버금과정과 함수"에 있는 바처럼), 이런 함수의 이름은 함수 인용으로 사용될 수 있다.

함수 인용의 사용은 연산 식의 연산항에만 한하지 않는다. 함수 인용 자체가 하나의 식이고 식이 허용되는 곳은 어디든지 사용될 수 있다. 이는 대입 문의 왼쪽과 같이 변수가 받아들이는 난이 되는 곳에는 사용되지 못한다. 그러나, 유사변수로 사용될 수 있는 3개의 내조립 함수가 있다. 유사변수(類似変数 Psedo-variable)란 받아 들이는 난으로 사용될 수 있는 내조립 함수의 이름이다.

보 기:

```
DECLARE   A   CHAR(10), B  CHAR(30);
SUBSTR(A,6,5) = SUBSTR(B,20,5);
```

이 대입 문에서, SUBSTR 내조립 함수는 보통의 함수 인용(오른쪽)과 유사변수 양쪽으로 사용되었다. 이의 뜻은 B에서 20번째 문자부터 5개 문자를 A의 6번째 자리부터 5개를 넣으라는 것이다.

제7절 자료 변환의 개념 (資料 変換의 概念)

자료 변환이란 (data conversion) 한가지 꼴로부터 다른 꼴로 값의 표현을 바꾸는 것이다. 만일 실행중에 변환이 일어나면, 실행시간을 늦추게 할 것이다.

조심하지 않으면, 변환은 정도의 손실을 초래하거나 예상할 수 없는 결과를 초래할 수도 있다. 이장은 우선 변환연산의 개념을 취급한다. 자세한 것은 제Ⅱ부, 제6장, "자료 변환"에 있다.

이 절은 변환의 과정을 취급한다. 변환에 관한 일은 두가지로 나누어 생각해 볼 수 있다. 첫째 목적 속성 (目的 属性 target attribute)을 결정하는 것이고 둘째 알려진 원시 자료와 목적 속성을 가지고 하는 변환 연산이다.

변환의 목적이란 변환된 값이 대입될 받아들이는 난이다.

보 기 :

```
DECLARE   A   PIC '$999 V.99', B  FIXED
DECIMAL (3,2),  C  FIXED BINARY(10);
A = B + C ;
```

식 B+C의 값내는 동안과 이 결과를 대입하는 동안, 네가지 틀리는 목적이 있다.

① B의 이진수 등가 (等価)가 대입될 편성자가 만든 임시 장소

② 더하기의 이진수 결과가 대입될 편성자가 만든 임시 장소

③ 이 이진수 결과의 십진수 고정점수 등가가 대입될 임시 장소

④ A, 결과의 마지막 목적지, 십진수의 수치 문자 등가가 대입된다.

첫번째 목적의 속성은 원시 (B)의 속성, 연산자, 상대쪽의 속성으로 부터 정해 진다. 둘째 목적의 속성은 원시의 속성 (C와 변환된 B의 속성)으로 부터 결정된다. 세번째 목적의 속성은 원시와 (둘째의 목적 속성) 마지막 목적 (A)의 속성으로 부터 결정된다. 네번째 목적 (A)의 속성은 DECLARE 분으로 부터 알려진다. 어떤 종류의 대행이 만들어지고 어떤 종류의 제약 (보기로, 최대 정도) 이 존재한다.

변환은 항상 원시 자료 항목과 목적 자료 항목을 포함한다. 이는 값의 처음 표현과 변환된 표현을 포함한다. 모든 원시 자료 항목과 목적 자료 항목의 속성은 편집 시 (時) 에 알려지거나 대행된다.

상수도 속성을 가짐을 알아야 한다; 상수 1.0은 상수 1, '1'B, '1', 1B, 1E0와 다르다. 상수는 편집시에 또는 실행시에 바꾸어지나, 규칙은 같다.

제 8 절 형식 변환의 목적 속성 (形式 変換의 目的 属性 target attributes of type conversion)

식의 연산항이 형식 변환을 요 (要) 할 때, 어떤 목적 속성 (targlt attribute) 이 대행되거나 원시 자료로 부터 추론되어야 한다. 이런 대행의 어떤 것은 표 4 - 1 에 보인 바 처럼 연산자에 기반을 둔다.

표 4 - 1　식의 연산항을 위한 목적 속성

연 산 자	목 적 . 형 식
+ - * / ** & | ¬ ||	규약 산수 비트 줄 문자 줄 (양쪽이 비트 줄이 아니면)
> < >= <= = ¬= ¬> ¬<	양쪽이 비트 줄이면 비트 줄 , 양쪽이 줄이면 문자 줄 , 아니면 산수 (지침은 양쪽이 지침이여야 하므로 변환이 없다.)

1 . 비트를 문자로와 문자를 비트로

이 변환에서 , 목적의 길이는 원시의 길이와 같다.

2 . 규약 산수를 비트 줄로

이 변환에서 목적의 길이는 원시의 정도로 부터 얻어진다. 목적의 길이를 결정하는 수식은 〃비트-줄 목적의 길이〃에 있다.

3 . 비트 줄을 규약 산수로

이 목적의 속성은 만일 최대 정도 (31) 의 고정점 이진수 고른수가 비트 줄 대신에 나타났다면 이 목적에 부여되는 속성이 된다.

조직 / 360 실무에서 고정점수로 바뀔 때 , 이 연산은 줄을 최대 정도의 고른수 (BINARY (31)) 로 먼저 변환함으로써 이루어 진다. 이 고른수가 목적 속성으로 변환된다.

〔1〕 산수 식 연산항을 위한 목적 속성

지수식을 빼고는, 산수 변환을 위한 목적 속성은 다음과 같이 실천된다.

BINARY 양쪽이 DECIMAL 인 경우는 제외, 양쪽이 DECIMAL 이면 기수 변환은 없다.

FLOAT 양쪽이 FIXED 인 경우는 제외, 양쪽이 FIXED 이면 척도 변환은 없다.

원시의 정도 기수나 척도 변환이 없을 경우는 제외 (표 4 - 2 , " 산수 변환을 위한 정도 " 를 보라)

지수식의 경우에, 기수와 정도는 다른 연산으로 결정된다. 첫째 연산항이 FIXED 이며 둘째 연산항이 부호없는 고정점수 고른수 상수이고 지수식의 결과가 허용된 최대 자리수를 넘지 않는 경우를 제하고는 첫째 연산항의 목적 척도는 항상 FLOAT 이다. (조직/360 실무에서, 십진수는 15 자리, 이진수는 31 자리의 고정점수이다)

둘째 연산항의 목적 척도는 고른수 상수나 정도 (P , 0) 의 고정점수 변수가 아니면 FLOAT 이다.

보 기 :

```
DECLARE
A FIXED DEC(2), B FIXED DEC(3,2), C FLOAT
DEC(4), D FLOAT DEC(7), E FIXED DEC(8),
F FIXED DEC (15);
```

D＊＊C 변환없음. 양쪽 연산항이 부동 점수

A＊＊4 변환없음. 둘째 연산항이 부호없는 고정점수 고른수 상수이고, 결과가 15 자리를 넘지 않을 것임.

D＊＊5　변환 없음. 첫째 연산항이 부동점수; 둘째는 정도 (P,0)인 고정점수.

D＊＊A　변환 없음. 첫째 연산항이 부동점수; 둘째는 정도 (P,0)인 고정점수

E＊＊A　둘째 연산항이 부호없는 고른수 상수 고정점수가 아니므로 첫째 연산항이 부동점수로 변환된다.
둘째 연산항은 정도(P,0)를 가지므로 변환되지 않는다.

D＊＊B　둘째 연산항은 정도(P,0)를 안가지므로 부동점수로 바뀐다. B가 비록 소수부분이 0인 고른수 값을 가지고 있어도, 이것의 선언된 정도가 (3,2)므로 이는 변환된다.

주 : D＊＊B를 뺀 모든 보기는 만약 이들이 십진이 아니고 이진으로 선언되었다 해도 이진수의 최대 자리수 31을 빼고는 같다. D＊＊B에서, B는 이진수이므로 척도 인자를 갖지 못하고, 만일 B가 정도 (3)을 가지면, 변환이 없다.

〔2〕 식 연산항 목적의 정도와 길이

다음 규칙은 모든 정도와 길이의 산출에 적용된다.

① 정도와 길이 지정은 언제나 고른수이다. 다음에 주어진 어떤 산출이 고른수 아닌값이 되면, 바로 다음의 큰 고른수가 결과 정도로서 취해진다.

보 기:

DECIMAL FIXED (8,3)을 BINARY FIXED로 바꾼다.

1+8＊3.32=27.56 결과의 자리수는 28.

3 * 3.32 = 9.96 결과의 척도 인자는 10

∴ (p , q) = (28 , 10)

척도 인자는 고정점 이진수로 변환에서 유지됨을 유의하라. 그러나, 만일 변환된 결과가 고정점 이진수 변수에 대입된다면, 소수의 이진수 자리는 고정점 이진수는 이를 위해 선언되는 소수가 없으므로 잘려나간다. 또한 척도 인자는 가끔 음수가 될 수 있다. (보기로, BINARY 와 DECIMAL 내조립 함수). 이런 경우, 절대값 (양의 수) 이 바로 다음의 큰 고른수를 취하는데 사용된다.

② 각 산수표현의 정도를 위한 최대치가 있다. 만일 어떤 산출이 이 한계보다 큰 값이 되면, 이 최대값이 대신 사용된다. 조직/360에서 이들 한계는 다음과 같다.

FIXED DECIMAL ..... 15 자리

FIXED BINARY ..... 31 자리

FLOAT DECIMAL ..... 16 자리

FLOAT BINARY ..... 53 자리

D - 편성자에서, 척도 인자 (소수) 는 고정점 십진 변수에서 0 ~ 15 이다. 이진 고정점 변수에서 지정 안되고 항상 0이 된다.

〔3〕 산수 변환을 위한 정도

표4 - 2는 만일 기수나 척도 변환이 일어나면 어떤 연산항을 위한 목적 정도를 보여준다.

식에 있는 연산항의 목적 속성은 다른 연산항의 정도에 영향을 받지 않는다.

표4－2． 산수 변환을 위한 정도

원 시 속 성	목 적 속 성	목 적 정 도
DECIMAL FIXED(p,q)	DECIMAL FLOAT	p
DECIMAL FIXED(p,q)	BINARY FIXED	1＋p＊3.32，q＊3.32
DECIMAL FIXED(p,q)	BINARY FLOAT	p＊3.32
DECIMAL FLOAT(p)	BINARY FLOAT	p＊3.32
BINARY FIXED(p,q)	BINARY FLOAT	p
BINARY FIXED(p,q)	DECIMAL FLOAT	p／3.32
BINARY FIXED(p,q)	DECIMAL FIXED	1＋p／3.32，q／3.32
BINARY FLOAT(p)	DECIMAL FLOAT	p／3.32

〔4〕 문자－줄 목적의 길이

만일 원시(原始)가 수치 문자 자료 항목이거나 비트 줄이며 목적이 문자 줄이면，목적의 길이는 원시의 길이와 같다.

〔5〕 비트－줄 목적의 길이

산수 연산항을 비트 줄로 변환할 때，산수 원시는 양의 이진수 고른수로 바뀐다. 이진 고른수 목적의 정도는 표4－3에 주어진 비트－줄 목적의 길이와 같다. 여기서 p－q는 고른수 부분의 자리수를 나타냄을 유의하라. D－편성자에서 목적 길이는 1～31 자리이다.

표4 - 3 . 비트-줄 목적의 길이

원 시 속 성	목 적 길 이
DECIMAL FIXED(p , q)	(p-q) * 3.32
DEIMAL FLOAT (p)	p * 3.32
BINARY FIXED (p , q)	p - q
BINARY FLOAT (p)	p

〔6〕 식의 값의 변환

완전히 값내진 식의 결과는 더 이상의 변환을 요할지 모른다.

이런 경우와 각 경우의 목적 속성이 표4 - 4에 있다. 덧붙여서 , 어떤 내조립 함수는 변환이 된다. 어떤 첨자 인용도 이진 고른수로 바뀐다.

표4-4. 변환이 일어날 수 있는 경우

원 인	목적 속성
대 입	등호의 왼쪽에 있는 변수의 속성
RETURN (식)	PROCEDURE나 ENTRY 문에서 지정된 속성

문	선택항	줄 길이
DISPLAY		원시, 최대 80문자
RECORD I/O	KEYFROM	ENVIRONMENT 속성에서 지정된 열의 길이
	KEY	ENVIRONMENT 속성에서 지정된 열의 길이 (또는 REGIONAL (1) 경우에서는 8문자)
STREAM I/O	STRING	서식 항목에서 지정된 줄 길이

문	선택항/속성	정 도
OPEN	PAGESIZE	8
I/O	SKIP	8
	LINE	8

제 9 절 변환 연산

목적 속성을 결정하는 경우처럼, 변환 연산은 두 단계로 분석해 볼 수 있다: 형식 변환과 (type conversion) 산수 변환 (arithmatic)

형식 변환에는 9개 경우가 있다.

- o 수치 문자를 문자-줄로
- o 수치 문자를 규약 산수로
- o 규약 산수를 수치 문자로
- o 규약 산수를 비트-줄로
- o 비트-줄을 규약 산수로
- o 문자-줄을 비트-줄로
- o 비트-줄을 문자-줄로
- o 수치 문자를 비트-줄로
- o 비트-줄을 수치 문자로

문자-줄을 산수로 그리고 산수를 문자-줄로 변환은 PL/I 서브세트에서 허용되지 않는다. 이것은 PUT STRING과 GET STRING 연산으로 목적을 달성할 수 있다; 그러나, 시간과 기억소를 많이 요함을 유의하라.

자세한 것은 제Ⅱ부, 제6장, "자료 변환"을 보라.

제 10 절 CONVERSION, SIZE, OVERFLOW, FIXEDOVERFLOW 조건

자료가 한 표현법에서 다른 것으로 바꾸어질 때, CONVERSION이나 SIZE 조건이 일어날 수 있다. OVERFLOW와 FIXED OVESFLOW

조건은 산수 연산의 결과가 실무상 정의된 한계를 넘을 때만 일어난다. 연산항이 한 표현에서 다른 것으로 변환될 때, 만일 결과의 값이 새로운 표현에 선언된 정도를 넘을 때는, SIZE조건이 일어난다.

SIZE 조건은 유의 숫자가 산수값의 왼편에서 떨어져나갈 때 일어난다. 이는 식 안에서 변환이 될 때나, 식의 결과를 대입할 때 일어날 수 있다. 이는 문자 줄이나 비트 줄로 변환될 때 값이 잘려나가도 일어나지 않는다. 이는 편집-지시 전송에서 E나 F 서식으로 변환될 때 지정된 난의 넓이가 나열 항목의 값을 수용하지 못하면 일어난다. SIZE 조건은 보통 불능하게 되어 있고, 만일 조건이 SIZE 조건 전치어의 범위 안에서 일어날 때만 중단이 일어난다.

CONVERSION 조건은 원시 난이 이루어질 변환에 대하여 무효한 문자를 포함할 때 일어난다. 보기로, 비트로 변환될 문자 줄이 0과 1이 아닌 문자를 포함하고 있을 때 CONVERSION 조건이 일어날 것이다. CONVERSION 조건은 보통 가능하게 되어 있다. 그래서 조건이 일어날 때, 중단이 일어난다. 이는 NOCONVERSION 전치어로 불능하게 될 수 있다. 이 경우 조건이 일어났을 때 중단이 일어나지 않는다.

OVERFLOW 와 FIXEDOVERFLOW는 실무 최대치(実務 最大値 implementation maximum)가 넘을 때 일어나고, 반면 SIZE는 선언된 정도(宣言된 精度 declared precision)가 넘을 때 일어남을 유의하라. OVERFLOW 조건은 F서식 항목에서 지정된 척도 인자가 너무 클 때 일어날 수 있다.

제 5 장 문의 분류 (文의 分類 statement classfication)

이 장은 문을 그들의 역할에 따라 분류한다. 각 기능 계급에 속하는 문이 나열되고, 각 문의 용도가 서술되고, 그들 사용의 보기가 예시 (例示) 된다.

제 1 절 문의 종류 (文의 種類)

문은 다음의 6종류로 묶어질 수 있다.

서술 (叙述 descriptive)

압력/출력 (入力/出力 input/output)

자료 이동과 계산 (資料 移動과 計算 data movement and computatonal)

통제 (統制 control)

예외 통제 (例外 統制 exception control)

프로그램 구조 (構造 program structure)

종류의 이름은 서술 목적으로만 정한 것이며, 그들이 언어에서 기본적 정의를 갖는 것은 아니다. 어떤 문은 하나 이상의 종류에 들어간다. 이는 그들이 둘 이상의 역할을 가질 수 있기 때문이다.

1. 서술 문 (descriptive statements)

PL/I 프로그램이 실행될 때, 여러가지 자료를 취급한다. 상수를 뺀 각개 자료 항목은 프로그램에서 이름에 의해서 인용된다. PL/I 언어는 인용되는 자료 항목의 특성 (또는 속성 (attribute)) 이

프로그램이 편집될 때 알려지는 것을 요구한다. 이 규칙에 하나의 예외가 있다. 어떤 화일에서는 INPUT나 OUTPUT 속성이 OPEN 문에서 지정될 수 있고, 그래서 프로그램의 실행중에 결정될 수 있다.

〔1〕 DECLARE 문

DECLARE 문은 이름의 속성〔属性 attribute〕을 지정하는(指定하는) 주된(主된) 수단이다. 완전한 속성의 조(組)가 지정되지 않았을 때 태만(怠慢 default)이 어느 이름에도 적용된다.

DECLARE 문은 다음에는 꼭 있어야 한다; 고정점 십진수와 부동점 이진수 변수, 문자와 비트 줄 변수, 화일이름, 지침 변수, 명찰(名札) 변수, 배열과 구조체, STATIC나 BASED 속성을 가진 자료, PICTURE 속성을 가진 자료, 일반으로 EXTERNAL 속성을 가진 자료, 한개의 프로그램 안에서 DECLARE 문의 수에는 제한없다.

〔2〕 그 밖의 서술 문

어떤 화일에 대해서는 OPEN문이 면의 크기(page sige) INPUT, OUTPUT 속성을 가질 수 있다. 고로 OPEN 문은 서술 문으로서 분류될 수 있다. FORMAT 문은 면이나 입력 카드처럼 외부 매체에 있는 양식을 서술하는 것으로 생각된다.

2. 입력/출력 문(INPUT/OUTPUT statement)

입력/출력의 주요 문은 내부 기억소와 외부 매체(媒体 medium)사이에 실제로 자료를 이동시키는 것들이다. 그 밖에 그런 이동에 영향을 끼치는 입력/출력 문은 입력/출력 통제 문(control statement)으로 생각된다.

다음에 열거한 것 중에서, 자료를 이동시키는 문은 두가지로 세분되고, 이는 RECORD I/O 와 STREAM I/O.

RECORD I/O 전송 문(転送 文 transfer statements)

READ, WRITE, REWRITE, LOCATE

STREAM I/O 전송 문

GET, PUT

I/O 통제 문

OPEN, CLOSE

이들 문과 함께 논술될 문은 DISPLAY 문이다.

STREAM 전송(転送 transmission)과 RECORD전송 간에는 상당한 차이가 있다. STREAM 전송에서 각 자료 항목은 별개로 취급되며, RECORD 전송은 전체로서 자료 항목의 집합체로(기록마디(record)) 간주된다. STREAM 전송에서 각 항목은 전송되면서 편집되거나 변환될 수 있다; RECORD 전송에서, 외부 매체에 있는 기록마디(記録마디 Record)는 편집이나 변환없이 내부 기억소에 있는 기록마디의 정확한 복사이다.

〔1〕 RECORD I/O 전송 문(transfer statement)

READ 문은 기록마디를 기억소로 직접 전송하거나 기록마디를 처리 가능하게 한다. WRITE 문은 출력 장치에 자료의 집합체를 전송하여 새 기록마디를 만든다. LOCATE 문은 또한 새 기록마디를 만든다. 그러나 이는 기록마디가 만들어질 완충역 자리를 사용할 수 있게 해준다.

REWRITE 문은 UPDATE 화일에 있는 기록마디를 교환한다.

〔2〕 STREAM 입/출 전송 문(入/出 転送 文 I/O transfer statements)

STREAM 전송 화일은 GET와 PUT 문을 가지고만 처리될 수 있는 순차 화일 (順次 화일 sequential file) 이다. 기록마디의 경계는 보통 무시된다 ; 자료는 외부 매체로부터 오거나 가거나 개개 자료 항목의 흐름 (stream) 이라 간주된다.

GET와 PUT 문은 두가지 형태의 하나로 항목의 나열을 전송한다 나열-지시 (羅列-指示 list - directed) 나 편집-지시 (編輯-指示 ldit - directed). 나열-지시 전송 (list - directed transmission) 에서, 자료는 외부에 빈자나 쉼표로 분리된 상수의 나열로서 기록된다. 편집-지시 전송 (edit - directed transmission) 에서, 자료는 외부에 서식 나열 (書式 羅列 format list) 에 따라 문자의 줄로서 기록된다.

주 : GET와 PUT 문은 화일 선택항 (option) 을 제거하고 STRINT 선택항을 지정하므로써 내부 자료 이동에 사용될 수 있다. 이에 대해서 "자료 이동과 계산문" 절에 논술된다. (제 9 장, "편집하기와 줄 취급하기" 에서도 이 분야를 언급한다)

〔3〕 입력/출력 통제 문 (入力/出力 統制 文 input/output control statements)

OPEN 문은 자료 집합 (資料 集合 data set) 을 가진 화일 이름과 연관되고 처리를 위해서 자료 집합을 준비한다. 이는 또 이 화일의 부가되는 속성을 지정한다. PRINT 속성을 가진 화일의 면 크기 (page sige) 는 OPEN 문에서만 지정될 수 있다. OPEN 문은 RECORD 전송 화일에는 항상 지정되어야 한다.

CLOSE 문은 화일로부터 자료 집합과의 연관을 끊는다. 모든 화일은 프로그램의 마침에 따라 닫힌다. 그래서 CLOSE 문은 항상

요하지는 않다.

〔4〕 DISPLAY 문

DISPLAY 문은 인쇄 전반 장치에 전언문(伝言文)을 쓸때 사용된다. 이는 또 REPLY 선택항(選択項 option)을 써서, 조작원(操作員 operation)이 부호나 전언문을 타자(打字)하므로써 프로그램에 연락하는데 쓰인다.

3. 자료 이동과 계산문(資料 移動과 計算文 data movement and computational statements)

외
2

내부의 자료 이동이란 지정된 변수에 식의 값을 대입하는 것이다. 식은 상수나 변수가 될 수 있고, 또한 해야할 산출(算出)을 지정하는 식이 될 수도 있다. 내부의 자료 이동에 가장 흔히 쓰이는 문은, 이는 또 계산 지정에도 많이 쓰임, 대입 문이다. 또 STRING 선택항을 가진 GET와 PUT 문이 내부의 자료 이동에 사용될 수 있다. 부가해서 PUT 문은 이루어져야 할 계산을 지정할 수 있다.

〔1〕 대입 문(代入 文 assignment statement)

대입 문은, 이는 열쇠말이 없음, 등호(=)에 의해서 지정된다 이는 다음 두가지 꼴 중의 하나를 취한다:

A = B ;

A = B + C ;

첫째 꼴은 순전히 내부의 자료 이동에 쓰일 수 있다. 등호의 오른쪽에 있는 변수(상수)의 값이 왼쪽의 변수에 대입될 것임을 말한다. 두번째 꼴은 등호의 왼쪽에 있는 변수에 대입될 연산 식을 가지고 있다. 둘째 꼴은 자료 이동과 함께 이루어져야할 연산

을 지정한다. 왼쪽에 있는 변수의 속성이 연산 결과의 속성과 달라도 되므로, 대입 문은 또 변환과 편집을 위해 사용될 수 있다.

왼쪽 변수는 배열이나 구조체의 이름이 될 수 있다; 왼쪽에 있는 식이 배열이나 구조체 값이 되어야 한다.

〔2〕 STRING 선택항

만일 STRING 선택항이 FILE 선택항 대신 GET나 PUT 문에 나타난다면, 이 문의 실행은 단지 내부의 자료 이동이 될것이다

보기 :

```
NAME은 30개 문자, FIRST, MIDDLE, LAST는 문자 줄
GET STRING(NAME) EDIT (FIRST, MIDDLE, LAST)
    (A(12), A(1), A(17));
```

이 문은 NAME의 처음 12개 문자가 FIRST에 대입되고, 다음 1개 문자가 MIDDLE에, 다음 17개 문자가 LAST에 대입됨을 지정한다.

```
PUT STRING(NAME) EDIT(FIRST, MIDDLE , LAST )
    (A(12), A(1) , A(17));
```

이 문은 FIRST, MIDDLE, LAST의 문자가 이 순서로 이어져서 NAME에 대입됨을 지정한다.

PUT 문에서는 계산 식이 사용될 수 있다.

보기 :

```
PUT STRING(BUFFER) EDIT(A*3, B+C)
    (F(15), F(15)) ;
```

이 문은 BUFFER에 대입될 문자 줄이 A*3, B+C의 값의 문자 표현으로 됨을 지정한다. 이런 방법으로 산수를 문자 줄로 문자 줄을 산수로 변환할 수 있다. 그러나 이는 시간과 기억소를

많이 요함을 유의하라.

4. 통제 문(統制 文 control statements)

PL/I 프로그램에 있는 문은 보통 통제의 흐름이 중단의 출현이나 다음 통제 문들 중의 하나의 실행으로써 변경되지 않는 한 순차로 실행된다.

GO TO(GOTO), IF, DO, CALL, RETURN, END, STOP

〔1〕 GO TO(GOTO) 문

GO TO는 무조건 건너뛰기(無条件 건너뛰기 unconditional branch)로 자주 사용된다. GO TO의 목적지가 명찰 변수로 지정되면, 명찰 상수를 대입하는 하나의 개폐기(스위치)로서 사용된다고 볼 수 있다. 목적지로서 명찰 변수를 가진 GOTO문은 활동 블러크(活動 블러크active block)안에 있는 문으로만 통제를 보낼 수 있다.

〔2〕 IF 문

IF 문은 대부분의 조건 건너뛰기(条件 건너뛰기 conditional branch)를 준비하고 보통 IF란 말 다음에 오는 단순 비교식과 함께 사용된다.

보기 :

```
IF   A = B
        THEN 참일 때 행위
        ELSE 거짓일 때 행위
```

만일 비교가 참(true)이면, THEN 절(節 clause)이 실행된다. THEN 절의 실행 다음에 통제는 ELSE 절을 넘어서 건너뛴다 그리고 실행은 계속된다. THEN 절은 GO TO 문이나 다른 통제 문을

가질 수 있다.

비교가 참이 아니면, 통제는 THEN 절을 넘어가서, ELSE 절이 실행된다.

보기 :

```
IF A=B THEN C=D ; ELSE C=E ;
```

THEN 절이 GO TO 문으로만 되어 있으면, ELSE 절은 없어도 되겠다.

```
IF A=B THEN GO TO LABEL ; 다음문
```

IF 열쇠말 (keyword) 다음에 오는 식은 요소 식만이 된다 ; 배열이나 구조체 변수는 요소 값을 돌려보내는 내조립 함수에 대한 인수 (引數 argument) 로서만 요소 식에 나타날 수 있다. 또 이 식은 하나 이상의 연산자를 가진 논리 식 (論理 式 logical expression) 이 된다.

보기 :

```
IF A=B & C=D THEN GO TO R ;
```

다음 셋은 등가이다.

```
IF A=B & C=D THEN GO TO R; B=B+1;
IF A=B THEN IF C=D THEN GO TO R;
B=B+1 ;
IF A¬=B THEN GO TO S ; IF C ¬ = D THEN GO TO R;
S : B=B+1;
```

〔3〕 DO 문

DO 문의 가장 많은 사용은 통제 변수 (統制 變数 control variable) 가 매 실행마다 증분되어서 정해진 번수만큼 실행될 문의 모임을 지정하는 것이다.

보기 :

DO I = 1 TO 10 ; . . . END ;

반복 실행될 문들은 DO 문으로 시작해서 END 문으로 끝나야 한다. 이 경우, 문들의 모임은 10번 실행될 것이다. 다음은 위와 등가이다.

I = 0 ; A : I = I + 1 ; IF I > 10 THEN GO TO B ;

. . . GO TO A ; B : 다음 문

증분은 통제 변수가 비교되기 전에 이루어지고, 보통은 통제 변수 값이 DO 문에 세워진 한계를 넘을 때만 통제는 모임 다음의 문으로 간다. 만일 마지막 반복이 끝난 후에 통제 변수를 인용하면, 변수의 값은 지정된 한계를 한개 증분(增分)을 넘을 것이다.

DO 문은 통제 변수없이 WHILE 선택항을 가지고 사용될 수 있다.

DO WHILE (A=B) ; . . . END ;

이는 A가 B와 같은 동안은 반복해서 실행된다.

둘을 함께 쓰면,

DO I = 1 TO 10 WHILE (A=B) ; . . . END;

이는 두개의 조건이 만족될 때만이 실행된다.

두개 이상의 계속되는 반복 지정(反復 指定 iteration specification)이 한 개의 DO 문에 들어갈 수 있다.

DO I=1 TO 10, 13 TO 15 WHILE (A = B) ; . . . END ;

DO I=1 TO 10, 13 TO 15 ; . . . END ;

첫번은 이대로 따른다면 최소한 10번을 시행하고, 다음에 A = B인 동안 3번 시행한다. 두번째는 13번을 시행한다.

DO 안에 있는 변수를 이용할 수도 있다.

DO I = 1 TO 10 ; A (I) = B + 1 ; , . . END;

별명이 없으면, 증분 (增分 increment) 이 1이다. 증분은 BY로 나타낸다.

DO I= 2 TO 10 BY 2 ; . . . END ;

통제 변수는 2 , 4 , 6 , 8 , 10으로 변한다.

〔4〕 반복 없는 DO 문

DO 문은 반복이 없을 수 있다.

DO ; . . . END;

이는 한번 시행이 된다. 이는 THEN이나 ELSE 절 안에 들어갈 수 있다.

〔5〕 CALL, RETURN , END 문

버금과정 (버금過程 subroutine) 은 버금과정의 입구점 (入口点 entry point) 이름이 붙은 CALL 문에 의해서 불러내진다. 통제는 RETURN 문이 버금과정에서 실행되거나 END 문의 실행이 버금과정을 끝낼때 활동시켰거나 불러낸 수속 (手続 procedure) 으로 돌아가게 된다.

괄호에 싸인 식을 가진 RETURN문은 함수 수속에서 값을 함수 인용 (function reference) 으로 돌려보내는데 사용된다. 이 형식은 함수 인용에 의해서 불러내진 수속으로 부터 돌아올 때만 사용된다.

프로그램의 정상 마침은 주 수속 (主 手続 main procedure) 의 마지막 END 문의 실행 결과로서나 주 수속의 RETURN 문의 실행 결과로서 일어난다. 이들 경우 통제를 운영 조직 (運営 組織 operating system) 으로 보낸다.

〔6〕 STOP 문

STOP 문은 프로그램을 비정상 마침 (非正常 마침 abnormal termination) 이 되게 한다.

5 . 예외 통제 문 (例外 統制 文 exception control statements)

앞 절에서 논술된 통제 문은 그들이 실행되면 통제의 흐름을 바꾼다. 실행의 순서가 바뀌질 수 있는 다른 방법은 예외 조건 (exceptional conalition) 의 출현으로 야기되는 프로그램 중단 (中斷 program interrupt) 이다.

보통, 예외 조건은 예상안한 행위의 출현이며, 넘치기 오류 (넘치기 誤謬 overflow error) 나 화일 끝 (end of fill) 과 같은 것이다 이런 조건의 처리에 대한 상세한 논술은 제 11 장, "예외 조건 처리와 프로그램 검토" 에 있다. 세개의 예외 통제 문은 ON, REVERT, SIGNAL 문이다.

〔1〕 ON 문

ON 문은 다음에 오는 지정된 조건의 출현이 프로그램 중단이 되게 할 때 취해지는 행위를 지정하는데 사용된다. ON 문은 어떤 조건에 대해서도 특정 행위 (行為 action) 를 지정할 수 있다.

이들 조건의 모두에 대해서, 표준 조직 행위 (標準 組織 行為 standard system ection) 가 PL/I의 일부로서 지정되어 있으며 중단 출현시 ON 문이 없으면, 이 표준 조직 행위가 취해진다. 대부분의 조건에서, 표준 조직 행위는 전언문을 인쇄하고 실행을 끝낸다.

ON 문의 서식 :

ON 조건-이름 { SYSTEM ; | on - 단위 }

〃조건 이름〃이란 제Ⅱ부, 제8장, 〃ON 조건〃에 열거된 열쇠말 중의 하나이다. 〃on - 단위는〃 그 조건이 일어나서 중단이 되었을 때 취해지는 프로그래머가 정의한 행위를 지정한다. 열쇠말 SYSTEM은 중단의 경우 표준 조직 행위를 지정하기 위해 on - 단위 대신에 사용된다.

보기 :

ON OVERFLOW GO TO OVA;

ON OVERFLOW;

ON OVERFLOW SYSTEM;

첫번째는 OVERFLOW에 의한 중단이 일어나면 OVA 명찰이 붙은 문으로 가라는것임. 두번째는 빈 문(빈 文 mull statement)이다. 이는 중단이 일어났을 때, 실행이 중단이 일어난 점부터 중단이 무시되어 계속하라는 것이다. 세번째는 중단이 일어나면 표준 조직 행위로 가라는 것이다. ON 문 자체가 없어도 표준 조직 행위에 맡기는 것이 된다.

on - 단위 (on - 単位 on - unit) 나 SYSTEM을 설정 (設定) 한 ON 문의 효력은 (1)같은 조건을 붙인 ON 문의 실행이나 (2) 그 조건의 이름을 붙인 REVERT 문의 실행에 의해서 한 블러크 안에서 변할 수 있다. 다른 블러크가 기동되었을(起動되었을) 때에 효력 있는 행위는 기동된 블러크로 보내지고 이 블러크에 의해 기동된 블러크에도 보내진다.

〔2〕 REVERT 문

이 REVERT 문은 REVERT 문이 나타난 블러크에서 실행

되는 동일 조건에 대한 모든 ON 문의 효력을 말소(末消)하는데 사용된다. 이의 효력도 ON 문과 같이 보내진다.

〔3〕 SIGNAL 문

SIGNAL 문은 명명된 조건으로 인한 중단의 출현을 가장한다(仮装한다)

보기:

ON OVERFLOW GO TO OVA ; . . .

REVERT OVERFLOW; . . . SIGNAL OVERFLOW ; . . .

위는 세가지 보기를 연관시킨 것이다.

6. 프로그램 구조 문(프로그램 構造 文 program structure statements)

프로그램 구조 문은 프로그램의 절(節)을 블러크와 모임으로(block와 group)구분하기 위해 사용되는 문들이다. 이들 문들은 PROCEDURE, END, ENTRY, BEGIN, DO 문이다. 블러크와 모임의 개념은 PL/I의 특성을 이해하는데 기본이 된다.

이는 제6, 7, 10장에서 자세히 취급된다.

프로그램을 블러크로 완전히 가르는 것은 프로그램을 짜고 시험하는 것을 간결하게 하며, 특히 여러 프로그래머가 한개의 프로그램을 짤때 그렇다. 이는 또 기억소의 효율적 사용 결과를 가져온다. 그것은 자동적 종류의 동적 기억소가 자료가 선언된 블러크에 들어갈 때 할당되기 때문이다.

〔1〕 PROCEDURE 문

PROCEDURE 문과 연관된 END 문으로 이루어지는 수속 블러크(procedure block)의 주된 역할은 지정된 자료에 이루어

지는 일련의 연산을 정의하는 것이다. 이 일련의 연산은 이름이 (PROCEDURE 문의 명찰) 주어지고 이 이름이 알려진 어떤 점에서도 불러내질 수 있다.

모든 프로그램은 적어도 하나의 PROCEDURE 문과 END 문을 가져야 한다. 하나의 프로그램은 서로 연락이 되며 분리되어 짜여진 여러개의 수속으로 구성될 수도 있다. 하나의 수속은 그 안에 품어진 다른 수속을 가질 수 있다.

수속으로 정의된 일련의 문은 수속 이름이 알려진 어느 점에서도 실행될 수 있다. 수속은 CALL 문에 의하거나 식에 이름이 나타나므로써 불러내진다. 두 수속 사이의 연락은 불러내는 수속에서 불러내지는 수속으로 보내지는 인수(引数 argument) 라는 수단으로 된다.

〔2〕 ENTRY 문

ENTRY 문은 이것이 나타난 수속에 대해 별도의 입구점 (entry point) 을 제공한다. 이는 인수 나열을 가진다.

주 : ENTRY 문은 이것이 나타난 수속에 대해 입구를 지정한다 ;
ENTRY 속성은 어떤 이름이 불러내질 수속임을 지정한다.

〔3〕 BEGIN 문

이름의 국지적 정의 (局地的 定義) 는 시작 블러크 (始作 블러크 begin block) 안에서 이루어질 수도 있다. 이 블러크는 BEGIN 문으로 시작되며 END 문으로 끝난다. 그러나, 시작 블러크는 한 프로그램의 정상 흐름으로 실행된다. 이는 자동 기억소가 할당되는 프로그램의 절 (節) 을 구분하는데 유용하다. 시작 블러크는 하나의 수속이나 다른 시작 블러크에 품어져야 한다.

〔4〕 DO 문

다른 종류의 프로그램 구조가 DO로 시작되며 END로 끝나는 DO - 모임에 의해 준비된다. DO - 모임은 그 안에 포함된 문이 통제 흐름의 목적을 위해 한개로서 간주됨을 지정한다. DO 문은 기준치가 만족될 때까지 일련의 문의 반복되는 실행을 지정해도 된다. 또 THEN 절(clause)이나 ELSE 절로 들어갈 수도 있다.

제 6 장 블러크, 통제의 흐름, 기억소 할당

(blocks, flow of control, Atorage allocation)

이 장은 PL/I 프로그램을 형성하기 위해서 문들이 어떻게 블러크로 조직되고, 한 프로그램 안에서 한 블러크에서 다른 것으로 통제가 흐르고, 기억소가 한 블러크 안에서 자료에 할당되는 가를 논술한다.

제 1 절 블러크 (blocks)

블러크란 프로그램의 한 절을 이루는 구분된 일련의 문이다. 이는 이 블러크 안에서 선언된 이름을 배치하고 변수의 할당을 한정한다. 두 가지의 블러크가 있다: 수속 블러크 (手続 블러크 procedure block) 와 시작 블러크 (始作 블러크 begin block) 이다.

1. 수속 블러크

수속 블러크란 간단히 수속이라 한다. PROCEDURE 문으로 시작해서 END 문으로 끝나는 일련의 문이다.

```
명찰: PROCEDURE ;
        . . .
      END [명찰];
```

모든 수속은 수속 이름이 통제가 옮겨가는 일차 입구점이기 때문에 이름이 붙어야 한다. PROCEDURE 문은 하나의 명찰만을 가져야 한다.

보기 :

PFADIN : PROCEDURE; . . . . END READIN;

PL/I 프로그램은 하나 이상의 이런 수속으로 이루어지고, 각개는 다른 수속과/또는 시작 블러크로 구성될 수 있다.

2 . 시작 블러크 (bagin blocks)

시작 블러크는 BEGIN 문으로 시작되고 END 문으로 끝나는 문의 집합이다.

〔명찰〕 . , . BEGIN;

. . .

END 〔명찰〕;

수속 블러크와 달라서, 명찰은 시작 블러크에서 선택적이다. 통제는 이름에 대한 인용없이 시작 블러크로 간다. 또 통제는 GO TO 문의 실행으로 명찰 붙은 BEGIN 문으로 간다.

보기 :

B : CONTROL : BEGIN; 문 - 1 문 - 2 . . . 문 - n END;

수속과 같지 않아서, 시작 블러크는 보통 특별한 인용을 통해서 통제가 주어지는 것이 아니다. 보통 문의 실행을 관리하는 정상 통제의 순서가 시작 블러크의 실행을 관리한다. 통제를 순차로 앞선 문의 실행 다음에 시작 블러크로 들어간다.

시작 블러크는 PL/I 프로그램의 구축에 불가결한 것은 아니다. 그러나, 프로그램의 어떤 영역을 구분하기 위해 시작 블러크를 이용하는 것이 유리할 때가 있다. 이에 대해서는 이 장과 제 7 장, "이름의 인지"에 논술된다.

3 . 내부와 외부 블러크 (內部와 外部 블러크 internal and external blocks)

어떤 블러크도 하나 이상의 블러크를 포함할 수 있다. 이는, 수속이나 시작 블러크나, 다른 수속이나 시작 블러크를 포함할 수 있다. 그러나, 블러크의 중첩은 안된다; 다른 블러크를 포함하고 있는 블러크는 이 블러크를 완전히 감싸야한다.

다른 블러크 안에 들어있는 수속 블러크를 내부 수속 (內部 手続 internal procedure) 이라 한다. 다른 블러크 안에 들어있지 않은 수속 블러크를 외부 수속 (外部 手続 external procedure)이라 한다. PL/I 프로그램에는 적어도 한개 외부 수속이 있어야 한다.

주 : 조직/360 실무에서, 각 외부 수속은 따로 편성된다.

시작 블러크는 언제나 내부이어야 한다. 이들은 언제나 다른 블러크 안에 들어가야 한다.

내부 수속과 시작 블러크는 품어진 블러크 (mested block) 라 할 수 있다. D - 편성자에서 허용된 최대 품기 수준은 3 이다. 외부 수속은 수준 (水準 level) 1 이다. 가장 밖의 블러크는 외부 수속이어야 한다.

보기 :

```
A :  PROCEDURE; 문 - a1 문 - a2 문 - a3
B :  BEGIN ;  문 - b1  문 - b2  문 - b3 END;
     문 - a4  문 - a5
C :  PROCEDURE; 문 - c1 문- c2
```

D : BEGIN; 문 - d1 문 - d2 문 - d3 문 - d4 END;
문 - a6 문 - a7 END;

위 보기에서, 수속 블러크 A는 다른 어떤 블러크에도 들어있지 않으므로 외부 수속이다. 블러크 B는 A에 들어있는 시작 블러크이다; 이는 다른 블러크를 포함하지 않는다. 블러크 C는 내부 수속이다; 이는 시작 블러크 D를 포함하고 있다. 이는 품기의 3 수준이다. A는 제 1 수준, B와C는 제 2 수준, D는 제 3 수준

주 : END는 항상 PROCEDURE, BEGIN, DO를 막아주어야 한다. END에 이름을 이을때는 당해 이름 (가장 가까운 PROCEDUR, BEGIN, DO 문의 이름) 이어야 한다.

제 2 절 블러크의 기동과 마침 (起動과 마침 activation and termination of blocks)

1. 기동 (起動 activation)

시작 블러크와 수속은 비슷한 외관과 기억소의 할당과 해방에서 같은 임무를 구사하고 이름의 범위를 구분하나, 이들은 기동되고 실행되는 방법에서 다르다. 시작 블러크는 단일 문처럼 정상 순서 프로그램 흐름의 과정에서 기동되고 실행된다. 그리고 보통, 단일 문이 나타날 수 있는 어디든지 나타날 수 있다. 수속에서, 정상 순서 프로그램 흐름은 수속을 피해간다. 수속이 기동될 수 있는 유일한 방법은 수속 인용 (手続 引用 procedure reference) 이다.

수속 인용은 다음 문맥의 하나에서 입구 이름 (入口 이름 entry

name) 이 나타나는 것이다.

① CALL 문에서 열쇠말 CALL 다음에

② 함수 인용 (函数 引用 function reference) 으로서 (제 10 장, "버금과정과 함수"에 자세히 있음)

이 장은 첫번째의 보기를 사용한다.

CALL 입구-이름;

입구 이름은 다음의 하나로 정의된다.

① PROCEDURE 문의 명찰

② 수속 안에 나타난 ENTRY 문의 명찰

이 중의 첫번째는 수속에 대해 일차 입구점 (一次 入口点 primary entry point)이라 불리고, 두번째는 이차 입구점 (二次 入口点 secondary entry point) 이라 불린다. (D-편성자에서, 외부 수속의 입구 이름은 6자를 넘지 못한다)

보기 :

```
A :      PROCEDURE; 문-1  문-2
FIRST :  ENTRY;  문-3  문-4  문-5
RETR :   ENTRY;  문-6  문-7  문-8  END;
```

이 보기에서, A는 일차 입구점이고 ERRT 와 RETR는 이차 입구점을 지정한다.

수속 인용이 실행되면, 지정된 입구점을 포함하고 있는 수속은 기동되고 불러내졌다 (invoked) 고 한다 ; 통제는 지정된 입구점으로 이행된다 (移行). 수속 인용이 있는 곳을 불러내는 점 (point of invocation) 이라 하고 인용이 있는 블러크를 불러내는 블러크 (invoking block) 라 한다. 수속이 일차 입구점에서 불러내지면 실행은 불러내진 수속의 첫번째 실행 가능한 문부터 시작한다. 수속이 이차 입구점에서 불러내지면, 실행은 ENTRY 문 다음의 첫번째

실행 가능한 문부터 시작한다. 정상 흐름에서 만난 ENTRY 문은 실행되지 않는다 ; 통제는 ENTRY 문을 피해간다.

어떤 수속도, 외부이든 내부이든, 언제나 외부 수속을 불러낼 수 있으나 어떤 다른 수속에 들어있는 내부 수속을 언제나 불러내지는 못한다. 포함하고 있는 수속에 상대적으로 첫번째 수준에 있는 내부 수속은 항상 포함하고 있는 수속에 의해 불러내질 수 있고, 또는 서로간에 불러내질 수 있다.

보기 :

```
PRMAIN :  PROCEDURE ;
          문-1  문-2  문-3
          A :  PROCEDURE ;
               문-a1  문-a2
          B :  PROCEDURE ;
               문-b1  문-b2
               END B;
          문-4  문-5
          C :  PROCEDURE;
               문-c1  문-c2
               END;
          문-6  문-7
          END;
```

이 보기에서 PRMAIN은 수속 A와C를 불러낼 수 있으나, B는 안된다 ; 수속 A는 수속 B와 C를 불러낼 수 있다 ; 수속 B는 수속 C를 불러낼 수 있다 ; 수속 C는 수속 A를 불러낼 수 있으나 B는 안된다. 수속은 활동중인 때는 불러내지 못한다.

프로그램은 운영 조직이 초기 수속(initial precedure 初期手続)을 불러낸다.

o 조직/360 실무에서, 이 수속을 주 수속(主手続 main procedure)이라 부른다.

o 이 주 수속은 이의 PROCEDURE 문이 OPTIONS (MAIN) 이 지정된 외부 수속이어야 한다.

보기 :

```
COTRL : PROCEDURE OPTIONS ( MAIN );
        DECLARE (A, B, C ) ENTRY;
        CALL A; CALL B; CALL C;
        END;
```

다음은 블러크의 기동에 대해서 기술되었거나 적어도 적용되는 것의 요약이다.

o 프로그램은 초기 수속이 운영 조직에 의해 기동될 때 활동적이 된다.

o 초기 수속을 제외하고는, 프로그램에 들어있는 외부 수속과 내부 수속은 이들이 수속 인용에 의해서 불러내질 때만 기동된다.

o 수속은 이것이 활동적일 때는 불러내지지 못한다.

o 시작 블러크는 정상 순서 흐름을 따라 기동된다.

o 초기 수속은 프로그램 기간동안 활동으로 남아 있다.

o 기동된 모든 블러크는 이것이 끝날 때까지 활동으로 남아있다 (다음을 보라)

2 . 마침 (termination)

보통 수속 블러크는 통제가 불러낸 블러크로 돌아가거나 다른 어떤 활동 블러크로 가면 끝마치게 된다. 유사하게 시작 블러크는 통제가 다른 활동 블러크로 갈 때 끝나게 된다.

〔1〕 시작 블러크 마침

시작 블러크는 다음의 어떤 하나가 일어나면 끝마치게 된다.

① 통제가 END 문에 닿는다. 이럴때 통제는 순차로 END 문 다음의 문으로 이동한다.

② 시작 블러크안에 있는 (또는 이 시작 블러크 안에 있는 것으로부터 기동된 블러크) GO TO 문의 실행이 통제를 이 블러크에 들어있지 않은 점으로 보낸다.

③ STOP 문이 실행된다 (실행을 끝마침)

④ 통제가 통제를 시작 블러크의 밖과 포함하고 있는 수속 밖으로 보내는 RETURN 문에 닿을 때.

제 2 항에서 서술된 형식의 GO TO 문은 다음과 같이 다른 블러크를 끝낸다.

만일 이행점 (移行点) 이 끝마치게될 블러크를 직접 기동시키는 블러크에 들어있지 않으면, 활동 도중의 사이에 낀 모든 블러크는 끝마치게 된다.

보기로, 만일 시작 블러크 B가 시작 블러크 A에 포함되었고 통제를 A에도 B에도 들어있지 않은 점으로 보내는 B의 GO TO 문은 A와 B를 다함께 끝낸다. 이 경우가 다음에 도시되었다.

```
FRST :  PROCEDURE OPTIONS(MAIN);
        문-1  문-2  문-3
    A :  BEGIN;
        문-a1  문-a2
    B :  BEGIN;
        문-b1  문-b2
        GO TO LAB;
        문-b3
        END;
        문-a3
        END;
        문-4  문-5
  LAB :  문-6  문-7
        END;
```

여기서 GO TO 문은 A와 B를 다 끝내게 한다.

〔2〕 수속 마침 (procedure termination)

수속은 다음의 하나가 일어나면 끝마치게 된다.

① 통제가 수속 안에 있는 RETURN 문에 닿는다. RETURN문의 실행은 통제를 불러낸 수속의 불러낸 점으로 돌려보낸다 만일 불러내는 점이 CALL 문이면, 불러내는 수속에서 실행은 CALL 다음의 문부터 다시 시작한다. 불러내는점이 함수 인용이면, 인용을 포함하고 있는 문의실행이 다시 시작될 것이다.

② 통제가 수속이 END 문에 닿는다. 효과에서 이는 RETURN 문의 실행과 같다.

③ 수속 안에 있는(또는 이 수속 안으로 부터 기동된 블러크

안에 있는) GO TO 문의 실행은 통제를 이 수속 안에 포함되어 있지 않은 점으로 보낸다.

④ STOP 문이 실행된다 (실행을 끝낸다)

항목 ①, ②, ③은 정상 수속 마침이다; 항목 ④는 비정상 수속 마침이다.

항목 ③은 시작 블러크의 항목 ③과 유사하다.

```
A :  PROCEDURE OPTIONS(MAIN);
     문-1  문-2
B :  BEGIN;
     문-b1  문-b2
     CALL C;
     문-b3
     END;
     문-3  문-4
C :  PROCEDURE;
     문-c1  문-c2  문-c3
D :  BEGIN;
     문-d1  문-d2
     END;
     문-c4
     END;
     문-5
LAB: 문-6  문-7
     END;
```

위 보기에서, 문 GO TO LAB 는 B,C,D를 끝나게 한다.

〔3〕 프로그램 마침 (program termcnation)

프로그램은 다음의 하나가 일어나면 끝마치게 된다.

① STOP 문이 프로그램 안의 어디서 실행된다. 이는 비정상 프로그램 마침이라고 하며, D-편성자에서, 이는 주 수속의 마지막 END 문으로 즉시 통제를 이행 (移行) 시킨다.

② 통제가 RETURN 문이나 주 수속의 마지막 END 문에 닿는다. 이를 정상 프로그램 마침이라고 한다.

③ ERROR 조건에 대한 빈 on-단위 (on-unit) 가 실행되거나 ERROR 조건에 대한 표준 조직 행위가 취해진다. 이 조건에 대한 표준 조직 행위는 통제를 운영 조직 통제 프로그램으로 돌려보내게 한다.

제 3 절 기억소 할당 (記憶所 割当 storage allocation)

기억소 할당 (storage allocation) 이란 변수로 표현되는 자료 항목 (들) 이 내부에 기록되게끔 기억소의 영역을 (領域) 변수와 연관시키는 방법이다. 기억소가 변수와 연관되었을 때, 변수는 할당되었다 (割当되었다 allocated) 고 한다. 주어진 변수의 할당은 프로그램 실행에 앞서 이루어지는 정적이거나 (静的이거나 statically) 실행 중에 이루어지는 동적으로 (動的으로 dynamically) 된다. 정적으로 할당된 변수는 프로그램 기간동안 할당된 그대로 남아있는다. 동적으로 할당된 변수는 그 변수를 포함하고 있는 블러크의 마침에 따라 또는 지침 취급 (指針 取扱 pointer manipulation) 에 의해서 변수의 기억소를 포기한다.

기억소가 변수에 대해 할당되는 방식은 그 변수의 기억소 종류 (種類 class) 에 따라 결정된다. 세가지 기억소 종류가 있다 : 정적 (靜的 static) , 자동적 (自動的 automatic) , 기초된 (基礎된 based) 각 기억소 종류는 이에 대응하는 기억소 종류 속성에 의해 지정된다 : 각각 STATIC, AUTOMATIC , BASED 이다. 마지막 둘은 동적 기억소 할당을 정의한다.

기억소 종류 속성은 요소 , 배열 , 대구조체 변수에 대해 명시로 선언될 수 있다. 만일 변수가 배열이나 대구조체 변수이면, 선언된 기억소 종류는 모든 요소에 적용된다.

기억소 종류 속성으로 명시 선언된 변수는 AUTOMATIC 속성을 가지는 것으로 대행된다. 여기에 예외가 있다 : EXTERNAL 속성을 가진 변수는 STATIC 속성을 갖게 대행된다.

1 . 정적 기억소 (靜的 記憶所 static storage)

STATIC 속성을 가진 모든 변수는 프로그램의 실행이 시작되기 전에 기억소에 할당되며 이들은 프로그램동안 할당된대로 남아있다.

보기 :

```
CNTRL :  PROCEDURE OPTIONS(MAIN);
         DECLARE(X,Y,Z) FIXED(5 , 0 ) STATIC EXTERNAL,
         ( OUTP, NEXT, REVERS ) ENTRY;
         X = 1;  Y= 1;  Z = 1;
         CALL OUTP; CALL NEXT; CALL REVERS;
         END; . . .
OUTP :   PROCEDURE;
         DECLARE X FIXED ( 5 , 0 ) STATIC EXTERNAL;
```

```
. . .
PUT EDIT('OUTP INVOCATION#', X)
(A(17), F(16)); . . .
X = X + 1;
END;
```

프로그램의 실행이 시작하기에 앞서, 모든 정적 변수는 할당된다. 그래서 위 보기에서, X, Y, Z는 초기 수속 CNTRL이 운영 조직에 의해 불러내지기 전에 할당된다. OUTP에 있는 X는 이 수속이 끝나도 값을 가지고 있다.

2. 자동적 기억소(自動的 記憶所 automatic storage)

AUTOMATIC 속성을 가진 변수는 변수가 선언된 블러크의 기동에 따라 기억소에 할당된다. 변수는 블러크가 활동적인 동안은 할당된 대로 남아있다; 이는 블러크가 끝나게될 때 해방된다. 한번 변수가 해방되면, 이것의 값을 잃게된다.

3. 기초된 기억소(基礎된 記憶所 based storage)

BASED 속성을 가진 변수는 기초된 변수로서 알려진다. 기초된 변수에 대한 변수는 SET 선택항을 가진 READ나 LOCATE 문을 사용하므로써 프로그래머에 의해 할당된다. 이는 기초된 변수의 서술이 지침 변수에 의해 "가르켜진" 기억역(記憶域)을 사용하는 방법으로 기초된 변수와 연관된 지침 변수에 의해 시작한다. 지침 변수는 기초된 변수의 서술이 다른 변수에 할당된 기억소와 중첩할 수 있게 하는 방법으로 (보기로, ADDR 내조립 함수로써) 의의를 가질 수 있다.

지침 변수는 기초된 변수의 서술이 다른 기억역에 적용될 수 있게끔 다루어질 수 있다. 이 제목에 대한 자세한 논술은 제12장, "기초된 변수와 지침 변수"에 있다.

제4절 서막과 종막(序幕과 終幕 prologues and epilogues)

블러크가 기동될 때마다 어떤 활동이 통제가 이 블러크에 있는 첫번째 실행 가능한 문에 닿기 전에 이루어져야 한다. 이런 활동의 집합을 서막(序幕 prologue)이라 한다. 비슷하게 블러크가 끝날 때, 어떤 활동이 통제가 이 블러크의 밖으로 이행(移行)되기 전에 이루어져야 한다. 이 활동의 집합을 종막(終幕 epilogue)이라 한다.

서막과 종막은 편성자의 할 일이지 프로그래머의 일은 아니다. 이들에 관한 지식이 프로그램의 수행을 개선하는데 프로그래머를 도울 수 있기 때문에 이들은 여기서 논술된다.

1. 서 막

서막은 논리상 블러크의 처음에 부속되었고 블러크의 활동에서 첫 단계로서 실행되는 편성자가 짠 과정(過程 routine)이다. 보통, 서막에 의해 이루어지는 활동은 다음과 같다.

- 자동적 변수를 위한 기억소의 할당
- on - 단위 (on - 単位 on - unit) 상속 (相続) 의 설정 (設定)
- 가인수 (仮引数 dumrny argument) 를 위한 기억소의 할당

2 . 종 막

종막은 논리상 블러크의 끝에 부속되었고 블러크의 마침에서 마지막 단계로서 실행되는 편성자가 짠 과정이다. 보통, 종막에 의해 이루어지는 활동은 다음과 같다.

o 블러크가 기동되기 전에 있었던 on - 단위 요건의 재설정 (再設定)

o 이 블러크에서 할당된 모든 자동적 변수를 위한 기억소의 해방

내 8

제 7 장 이름의 인지 (이름의 認知 recognition of names)

PL/I의 프로그램은 표식어, 상수, 연산자나 구분자로 사용되는 특수 문자의 집합체로 이루어진다. 표식어는 열쇠말이거나 프로그래머에 의해 지정된 뜻을 가지는 이름이다.

PL/I언어는 편성자가 보통 문맥으로 부터 표식어가 열쇠말인지 아닌지를 결정할 수 있게끔 만들어졌다. 그래서 프로그래머가 정의한 이름으로 사용되지 못하는 예약된 말 (予約된 말 reserved word) 이 매우 적다. 어떤 표식어도 이름으로 사용해도 좋다; 단 하나의 제한은 프로그램의 어디에 있던지 이름은 오직 한가지 뜻만을 가질수 있다는 것이다. 보기로, 같은 이름이 화일과 부동점수 변수에 함께 사용될 수 없다.

주 : 48 - 문자 조 연산 표식어 GT, GE, NE, LE, LT, NG, NL, NOT OR, AND, CAT는 48 - 문자 조가 사용되면 예약된다. 다른 이유로 해서, 48 - 문자 조가 사용되면 PT도 예약된 말이다. (PL/I 언어는 부분적으로 표식어 SYSIN 과 SYSPRINT를 예약한다. 그러나 D - 편성자는 안한다. 그러나, 만일 프로그래머가 이들 표식어를 D - 편성자에서 정의된 표준 조직 화일과 연관시키면 주의를 요한다. 이는 제 8 장, "표준 화일"에 자세히 논거된다)

그러나 한개 이름에 프로그램 전체를 통해 동일 의미를 가질 필요는 없다. 한 블러크 안에서 선언된 이름은 오직 이 블러크 안에서만 의미를 가진다. 블러크 밖에서는 이것이 동일 이름이 이

블러크 밖에서도 선언되지 않는한 알려지지 않는다. 이 경우, 밖의 블러크에 있는 이름은 다른 목적으로 인용된다. 이는 국지적 정의를 지정하고, 그래서 프로그램의 다른 부분을 짜는 다른 프로그래머에 의해 사용되는 이름을 알지 않아도 수속이나 시작 블러크를 짤 수 있게 해준다.

뜻이 통용되는 프로그램 부분을 그 이름의 선언 범위(宣言 範囲 scope of the declaration)라고 부른다. 대부분의 경우, 이름의 범위는 순전히 이름이 선언된 위치에 의해서 결정된다.

이름의 범위에 관한 규칙을 알기 위해서는 〃~에 포함되어 있다〃와 〃~에 내적이다〃라는 용어를 이해할 필요가 있다.

~에 포함되어 있다:

블러크의 모든 원문(原文)은, PROCEDURE나 BEGIN 문에서 관련된 END 문까지, 이 블러크에 포함되어 있다(contained in)고 한다. 그러나 블러크의 앞에 오는 BEGIN이나 PROCEDURE 문의 명찰은, ENTRY 문의 명찰과 같이, 이 블러크에 포함되어 있지 않다. 품어진 블러크(nested block)는 이것이 나타난 블러크에 포함되어 있다.

~에 내적(內的)이다:

한 블러크에 포함되어 있으며, 이 안에 품어진 어떤 다른 블러크에도 포함되어 있지 않은 원문(原文)을 이 블러크에 대해 내적(內的 internal to)라고 한다. 수속의 입구 이름은(또는 BEGIN 문의 명찰) 이 블러크에 포함되어 있지 않음을 주목하라. 결과로, 이들은 포함한 블러크에 내적이다. 외부 수속의 입구 이름은 D-편성자에서 외부 수속에 내적인 것으로 취급된다.

이들 용어에 추가해서 선언의 여러가지 형식은 중요하다. 세가지 형식--명시 선언(明示 宣言 explicit declaration), 문맥상 선언(文脈上 宣言 contextual declaration), 암시 선언(暗示 宣言 inplicit declaration)은 다음 절에서 논술된다.

제 1 절 명시 선언

이름이 만약 다음 중에 나타나면 명시 선언된다.

① DECLARE 문에서

② 매개변수 나열에서

③ 문 명찰로서

④ PROCEDURE나 ENTRY 문의 명찰로서

매개변수 나열에서 이름의 출현은 그 이름을 위한 DECLARE 문이 매개변수 나열이 나타난 PROCEDURE 문 바로 다음에 나타난 것과 같다. 같은 이름이 이 블러크에 내적으로 DECLARE 문에서 나타나도 된다.

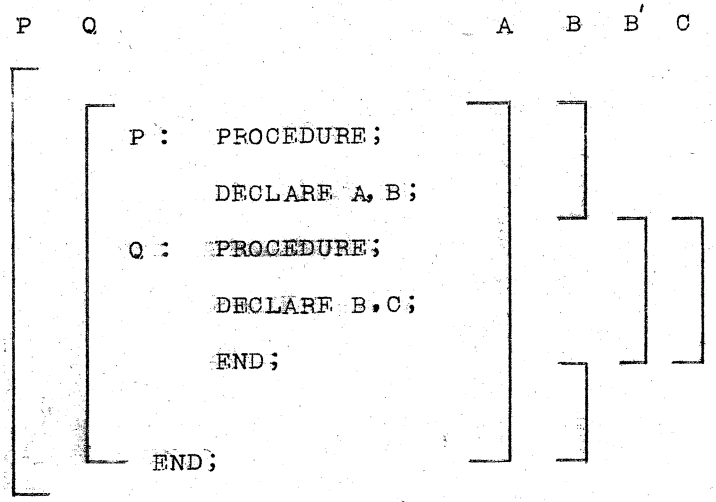
내부 PROCEDURE나 ENTRY 문으로서 이름의 출현은 이 이름이 이것이 인용되는 수속에서 PROCEDURE 문 바로 앞에서 DECLARE 문에 선언된 것과 같은 효과를 가진다. 외부 수속의 PROCEDURE와 ENTRY 문의 명찰은 D-편성자에서 이들이 외부 수속에서 EXTERNAL 수속을 가지고 DECLARE 문에 나타난 것처럼 취급된다

문 명찰 전치어의 출현은 이 블러크에 내적인 DECLARE 문에서 선언되는 변수와 같게 명시 선언이 된다.

1. 명시 선언의 범위 (範囲 scope of an explicit declaration)

이름의 명시 선언 범위는 선언이 내적인 블러크이고, 동일 표식어에 관한 다른 명시 선언이 내적인 모든 포함되어 있는 블러크를 제외한다.

보기:



왼편의 대괄호는 명찰 P와 Q의 범위를 보인다. 오른편의 대괄호는 이름의 범위를 보인다. B와 B'는 이름 B의 별도 용법을 가르킨다.

제 2 절 문맥상 선언

표식어가 어떤 문맥에 나타나면, 이는 그 표식어에 관한 명시 선언 없이도 인지될 수 있다. 이런 표식어는 만일 이것이 이와 같은 표식어를 위한 명시 선언의 범위 안에 놓여있지 않다면 문맥으로 선언되었다고 한다.

명시 선언되지 않은 표식어는 만일 다음 중의 하나라면 BUILT-IN으로서 문맥상 선언되고 인지된다.

〔1〕 표식어가 CALL 문에서 열쇠말 CALL 바로 다음에 따라온다.

〔2〕 표식어가 식이 있을 수 있는 문맥에서 괄호에 싸인 나열을 뒤에 달고있다; 이는 표식어가 인수를 가진 함수 인용에서 함수 인용으로 나타났을 때이다.

주: 만일 현행 D-수준 편성자의 실무에서, D-편성자에 내조립 함수나 유사변수로 알려진 그 밖의 표식어가 위 문맥의 하나에 나타나면, 이는 BUILTIN 속성 대신에 ENTRY 속성을 받는다 그리고 경고 선언문(warning message)이 인쇄된다. 그러나 그런 표식어는 ENTRY 속성을 가지고 명시 선언될 것을 권장한다.

1. 문맥상 선언의 범위 (文脈上 宣言의 範囲 scope of a contextual declaration)

문맥상 선언의 범위는 선언이 이름이 나타난 외부 수속의 PROCEDURE 문 바로 다음에 오는 DECLARE 문에서 된 것처럼 결정된다.

문맥상 선언은 이름이 외부 수속에서 선언된 것과 같은 효과를 가지며, 이 문이 외부 수속에 포함되어 있는 블러크에 내적일 때라도 그러함을 유의하라. 그 결과로서, 이름은 온 외부 수속 전반에 알려지고, 이름이 명시 선언된 블러크는 제외된다. 문맥상 선언은 명시 선언의 범위 안에 존재할 수 없으므로, 표식어의 문맥을 위해서 명시 선언된 그 표식어를 위해 설정된 속성에 추가하는 것은 불가능하다.

제 3 절 암시 선언 (暗示 宣言 implicit declaration)

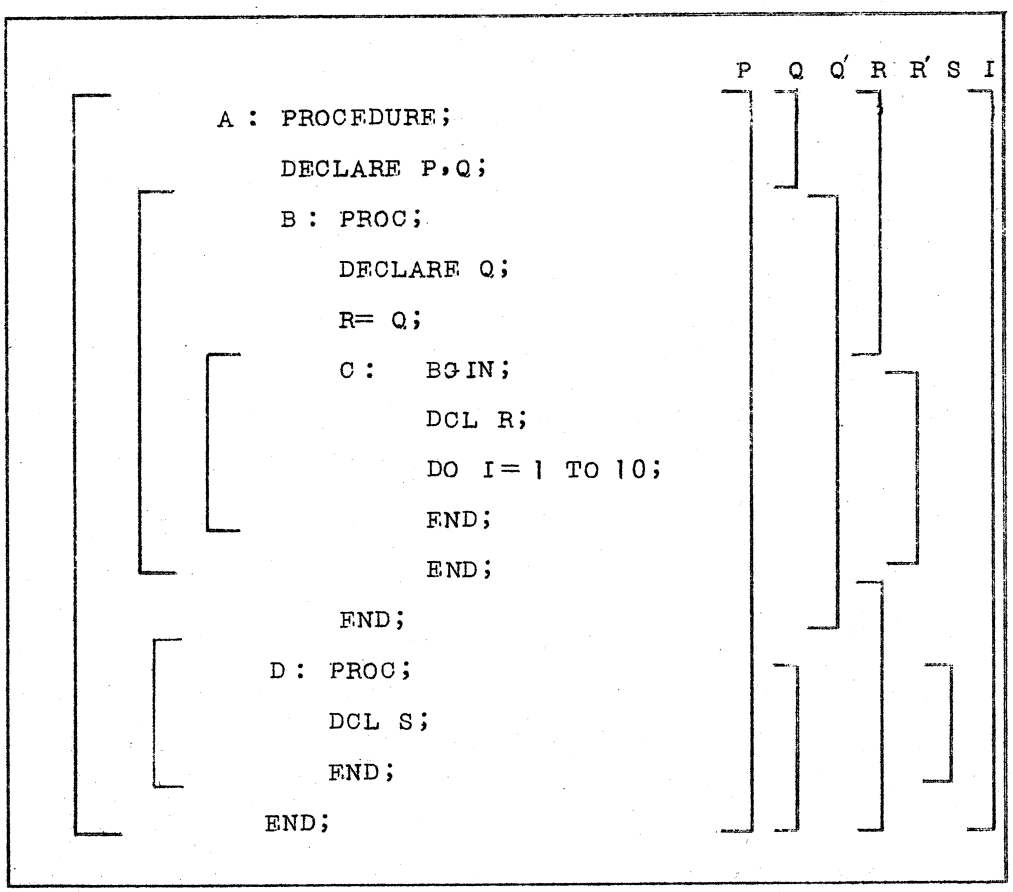
만일 이름이 프로그램이 나타나고 명시로나 문맥상으로나 선언되지 않으면, 이를 암시 선언되었다고 한다. 암시 선언의 범위는 이 이름이 사용된 외부 수속의 첫번째 PROCEDURE 문 바로 다음에서 DECLARE 문에서 선언된 것처럼 결정된다.

암시 선언은 이름의 첫째 문자에 따라 태만 속성 (怠慢 属性 default attribute) 을 적용하게 한다. 만약 이름이 I 부터 N 사이의 문자로 시작하면 속성 FIXED BINARY(15) 를 받는다. 이름이 다른 문자 (확대 영문자포함) 로 시작하면 속성 FLOAT DECIMAL(6)를 받는다. (태만 정도는 조직/360 실무에서 정의된 것들이다) 표식어 TIME,DATE,NULL 은 이들이 해당 내조립 함수를 인용하지 않는한 암시 선언될 수 있다. 이 경우 이들은 BUILTIN속성을 가지고 명시 선언되어야 한다.

주 : 현행 D - 수준 편성자의 실무에서, TIME, DATE, NULL 은 항상 명시 선언되어야 한다. 만일 이들이 명시 선언되지 않으면, 해당 내조립 함수에 대한 인용으로 대행되고 경고 전언문이 인쇄된다.

제 4 절 선언의 보기

자료 선언의 범위가 도해 7 - 1 에 설명되었다. 왼편 대괄호는 블러크 구조를 가르키고, 오른편 대괄호는 각개 이름의 선언 범위를 보인다. 도해에서 Q 와 R 의 두가지 선언 범위는 Q 와 Q', R 과 R' 로 보인다.



도해 7 - 1 자료 선언의 범위

P는 블러크 A에서 선언되었고 이것은 재선언되지 않았으므로 A 전반에 알려진다.

Q는 A에서 선언되었다. 그리고 B에서 재선언되었다. 첫번째 선언의 범위는 B를 뺀 전체 A이다. 두번째 선언의 범위는 블러크 B만이다

R은 블러크 C에서 선언되었다. 그러나 R에 대한 인용이 블러크 B에서도 되었다. 블러크 B에서 R에 대한 인용은 외부 수속 A에서 R에 대한 암시 선언이 된다. 그러므로 서로 다른 범위를

가진 독립된 두개의 이름이 존재한다. 명시 선언 R의 범위는 C이다; 암시 선언 R의 범위는 블러크 C를 뺀 전체 A이다.

S는 수속 D에서 선언되었고 D에만 알려진다.

I는 블러크 C에서 인용되었다. 이는 외부 수속 A에서 암시 선언이 된다. 그 결과로 이 선언은 포함된 수속 B, C, D를 넣어서 전체 A에 적용된다.

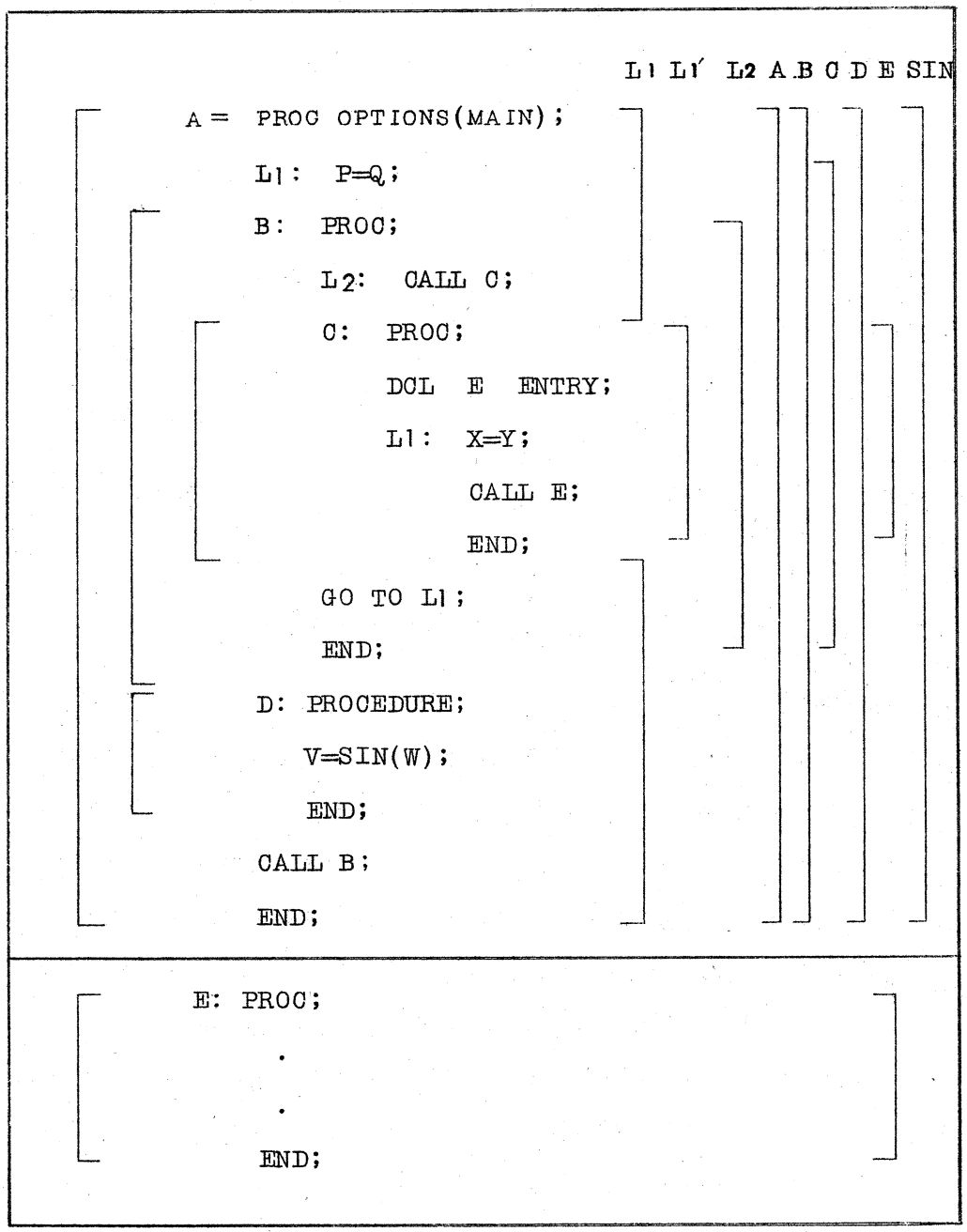
입구 이름, 문 명찰, 문맥상 선언의 범위는 도해7-2에 도시되었다. 보기는 두개의 외부 수속을 보인다. 이들 수속의 이름은 A와 E, 이들이 사용된 수속안에서 EXTERNAL 속성을 가지고 명시 선언된 것으로 간주된다. 덧붙여, E는 입구 이름으로서 블러크 C에서 명시되었고 태만에 의해 EXTERNAL 속성을 받는다. 이름 E의 범위는 블러크 C의 전체와 블러크 E의 전체이다. 이름 A의 범위는 블러크 A의 전체만이다. 복귀는 허용치 않으므로, A는 E 안으로 부터 불러내지지 못한다; 고로 A는 E 안에 알려지지 않는다.

명찰(名札 label) L1은 A와 C에 내적인 문으로 나타났다. 고로 두개 분리된 선언이 설정된다; 첫번째는 블러크 C를 뺀 블러크 A 전체에 적용된다. 두번째는 블러크 C에만 적용된다.

그러므로, 블러크 B에서 GO TO 문이 설행되면, 통제는 블러크 A에 있는 L1으로 이행(移行)된다. 그리고 B는 끝마친다.

D와 B는 블러크 A에서 명시되었고 A 안 어디서나 인용될 수 있다; 그러나 이들은 INTERNAL이므로, 이들은 블러크 E에서 인용되지 못한다. (인수로서 E로 보내지지 않는한).

C는 B에서 명시되었고 B에서 인용될 수 있다. 그러나 B 밖에서는 안된다.



도해 7 - 2 입구, 명찰, 문맥상 선언의 범위

L2 는 B에서 선언되었고 블러크 B에서 인용될 수 있고, B에 포함되어 있는 C에서도 같으나, B 밖에서는 안된다.

이름 SIN은 BUILTIN으로 문맥상 선언되었다. SIN의 범위는 문맥상 선언이 되는 문이 블러크 D에 있지마는 블러크 A 전체이다.

제5절 태만 속성의 응용

이름과 연관된 속성은 명시적, 문맥상, 암시적 선언된 것과 태만에 의한 대행 (代行)된 것으로 구성된다. 각개 속성에 관한 태만은 제Ⅱ부, 제9장, "속성"에 있다.

제6절 INTERNAL과 EXTERNAL 속성

INTERNAL 속성을 가진 이름의 범위는 이것의 선언 범위와 같다. 이 이름에 관한 다른 어떤 선언도 다른 속성을 가진 새 목적으로 인용되며, 범위가 중첩되지 않는다.

EXTERNAL 속성을 가진 이름은 다른 외부 수속에서나 외부 수속에 포함되어 있는 블러크 안에서나 한 프로그램에서 두번 이상 선언되어도 된다. 이 이름의 각개 선언은 하나의 범위를 설정한다. 이들 선언은 서로 연락이 되며, EXTERNAL 속성을 가진 동일 표식어에 관한 모든 선언은 동일 이름을 인용한다. 이 이름의 범위는 이 이름에 관한 모든 선언 범위의 합친 것이 된다.

이들 선언 모두가 한가지를 인용하므로, 이들은 동일한 속성 조가 되도록 해야 한다. 편성자가 이를 검사하지 못하므로, 주의를 요한다. (INITIAL 속성을 포함해서). 편성시 나올 수 있는 속

성을 나열한 것은 이름의 사용을 검사하는데 도움이 될 것이다.

D-편성자는 EXTERNAL 속성을 가진 이름을 6 문자 이하로 제한한다. 이는 화일 이름과 외부 수속의 입구 이름과 같이, 태만에 의한 EXTERNAL 이름도 마찬가지다.

보기 : 다음 보기는 이 장에서 논술된 점을 도시한다.

```
A : PROCEDURE OPTIONS(MAIN);
    DCL S CHAR(10), (SET,OUT) ENTRY;
    CALL SET (23168);
E : GET EDIT . . . ;
    B : BEGIN;
    DCL(X,Y) DEC;
    GET EDIT(X,Y,N) . . . ;
    CALL C(X,Y);
    C : PROC (P,Q);
        DCL S BIN EXTERNAL ; . . .
        GET EDIT(I) . . . ;
        IF . . . THEN GO TO B;
        CALL D(I); CALL OUT(E);
    B : END C;
    D : PROC(N);
        PUT EDIT(N,S) . . . ;
        END D;
    END B;
  GO TO E;
  ENE A;
```

```
OUT: PROC(R);
     DCL R LABEL, S BIN EXTERNAL, Z DEC FIXED,(M,L)
     STATIC INTERNAL ; . . . .
     GO TO R;
SET: ENTRY(Z);
     X=Z;
     RETURN;
     END OUT;
```

A는 외부 수속 이름이다; 이의 범위는 블러크 A 전체다.

S는 블러크 A와 C에서 명시 선언되었다. 이 문자 줄 선언은 블러크 C를 뺀 블러크 A 전체에 적용된다; 이진수 선언은 블러크 C 안에만 적용된다. D가 블러크 C에서 불러내질 수 있지만, D에 있는 PUT문에서 S의 인용은 문자 줄 S에 관한 인용이고, 블러크 C에서 선언된 S에 관한 것이 아니다.

N은 블러크 D에서 매개변수로서 나타난다. 그러나 이는 또한 이 블러크 밖에서도 사용된다. 이것이 매개변수로 나타나면 D 안에서 N의 명시 선언이 설정되고, D 밖에서 이를 인용하면 블러크 A에서 N의 암시 선언이 된다. 이들 두 N의 사용은 다른 목적의 인용이다. 이 경우 이들은 동일자료 속성을 가진다.

X와 Y는 B 전반에 알려지고 블러크 C에서나 D에서 인용될 수 있다. 그러나 B 밖의 A에서는 안된다.

입구점 SET 아래에 사용된 X는 OUT 안에서 암시 선언이 되고 OUT 밖에서는 알려지지 않는다.

P와 Q는 매개변수이다; 매개변수 나열에 나타난 것으로 명시

선언이 되기에 충분하다.

I는 외부 수속 A에서 명시 선언되지 않았다; 이는 암시 선언되고 A전반에 알려진다. 그러나 이는 블러크 C에만 나타났다.

외부 수속 A에서, OUT와 SET는 입구 이름으로 명시 선언되었다. 그리고 태만에 의해 EXTERNAL 속성을 받는다.

보기에 있는 두번째 수속은 두개의 입구 이름을 갖는다. SET와 OUT. 이들은 EXTERNAL 속성을 가지고 명시 선언된 것으로 간주된다. 이들 두 입구 이름 SET와 OUT는 두개의 외부 수속 전반에 알려진다.

명찰 B는 보기에서 두번 나타났다. A에서 명찰로서 명시 선언된, 시작 블러크의 명찰로 한번 나타났다. 이는 블러크 C에서 END문의 전치어로 나타나서 재선언되었다. 블러크 안 GO TO 문에서 B를 인용한 것은 END 문의 명찰을 인용한 것이 된다. 블러크 C 밖에서, B에 관한 어떤 인용도 시작 블러크의 명찰에 관한 것이된다.

C와 D를 B 안 어느 점에서도 부를 수 있으나 B 밖의 A 부분과 다른 외부 수속에서는 안됨을 유의하라.
비슷하게, E가 A 전반에 알려지므로, E로 옮겨가는 것은 A 안 어디서도 가능하다.

그러므로 품어진 블러크의 밖으로 옮겨가는 것은 가능하나, 보통 그런 블러크 안으로 옮겨가는 것은 안된다.

위 규칙에 관한 예외가 외부 수속 OUT에서 발견된다. 여기서 블러크 A에 있는 명찰이 인수로서 명찰 매개 변수 R로 보내

진다.

문 GO TO R은 통제를 명찰 R로 보낸다. (이에 관련된것은 제10장, "버금과정과 함수"에서 상세히 논술된다.)

변수 M과 L은 블러크 C에서 STATIC로 선언되었다. 그래서 값이 보존된다.

OUT 에 있는 S와 C에 있는 S를 동일한 것으로 하려면, 이들은 다같이 속성 EXTERNAL을 가지고 선언되어야 한다.

제7절 중복 선언과 모호한 인용

동일 블러크에 내적인 동일 표식어에 관한 둘 이상의 선언은 중

중복 선언(重複宣言 multiple declaration)이 된다.

수식된 이름을 만들 수 있는 경우는 예외이다.

EXTERNAL 속성을 가지며 상이한 자료 속성을 가진 동일 표식어에 대한 둘이상의 선언은 중복 선언이 된다.

중복 선언은 오류(誤謬)이다.

수식된 이름에서, 수식하는 첫번째 이름은 고유해야 한다. 어떤 이름에 관한 인용은 항상 이 인용이 있는 가장 안쪽 블러크에서 선언된 표식어에 적용되도록 취해진다.

모호한 인용이란(ambiguores reference)이름을 고유하게 만드는데 불충분한 수식을 가진 이름이다.

보 기 :

```
DCL 1 A, 2 C, 2 D, 3 E;
BEGIN;
DCL 1 A, 2 B, 3 C, 3 E;
A.C=D.E;
```

여기서 A.C는 안쪽 블러크 C에 있는 블러크를 인용한다; D.E는 바깥쪽 블러크에 있는 E를 인용한다.

```
DCL 1 A, 2 B, 2 C, 3 D, 2 D;
```

이 보기에서 ; B는 이중으로 선언되었다. A.D는 두번째 D를 인용한다.

A.D는 두번째 D만의 완전한 수식이다; 첫번째 D는 A.C.D로 인용될 수 있을 것이다.

```
DCL 1 A, 2 B, 3 C, 2 D, 3 C;
```

이 보기에서, A.C는 어느쪽 C도 이것으로 완전히 수식되지 않으므로 모호하다.

D-편성자에서, 수식한 것의 첫번째 이름은 고유해야 한다.

보 기 :

```
DECLARE 1 ATR, 2 A1, 3 B1, 3 B2,
          4 D1, 4 D2, 2 A2, 3 B1,
```

4 D3, 4 D4, 3 B3;

수식한 것 B1.D3는 B1이 고유하지 않으므로 허용되지 않는다.

제 8 장 입력과 출력 (入力과 出力
input and output)

PL/I은 자료를 계산기의 내외 (內外) 기억장치 (storage device) 간에 전송시키는 입력과 출력 문을 갖추고 있다. 프로그램에 외적인 자료의 집합체를 자료 집합 (資料 集合 data set) 이라 부른다. 자료 집합에서 프로그램으로 자료를 전송 (転送 transmission) 하는 것을 입력 (入力 input) 라 부르고, 프로그램으로부터 자료를 전송하는 것을 출력 (出力 output) 이라 부른다.

자료 집합은 천공 카-드, 자기 테이프 타래, 자기 원반, 자기 자료 각과 같은 여러가지의 외부 기억 매체 (外部 記憶 媒体 external storage media) 에 기억된다. 이렇게 다양함에 불구하고, 외부 기억 매체는 자료를 모으고, 저장하고, 전송하는데 표준방법을 사용할 수 있는 매우 공통된 성질을 가지고 있다.

외 1 순

약속으로 일반 용어 "권" (巻 volume) 이란 것이 자기 테이프 타래, 원반, 자료 각 (data cell) 과 같은 것들이 이들의 특정한 물리적 구조를 고려하지 않고 외부 기억소의 한 단위를 인용하는데 사용된다. 자료 집합에 있는 자료 항목은 블러크 (block) 라 불리는 독특한 물리적 모임으로 배열된다. 이들 블러크들은 하나의 단위로서 보다는 부분적으로 자료 집합을 전송하고 처리하게끔 허용한다. 처리 목적을 위해서, 각 블러크는 기록마디 (記録마디) 라고 불리는 하나 이상의 소부분 (小部分) 으로 구성되고, 그 것은 하나 이상의 자료 항목을 포함할 수 있다.

블러크는 또 물리 기록마디 (物理 기록마디 physical record) 라 불린다. 이유는 이것이 권 (巻) 으로 또는 권으로 부터 물리적으로

전송되는 자료의 단위이기 때문이다. 물리 기록마디와 이것의 논리적 소부분의 혼동을 막기 위하여, 논리적 소부분을 논리 기록마디(論理 記錄마디 logical record)라 부른다.

블러크가 둘 이상의 논리 기록마디를 포함하고 있으면, 이들 기록마디들은 블러크되었다(blocked)고 한다. 블러크된 기록마디는 기억소의 사용을 효과있고 손실 없게 해준다.

제 1 절 자료 전송의 형식(資料 轉送 形式 types of data transmission)

두가 형식의 자료 전송이 PL/I 프로그램에서 사용될 수 있고, 이는 흐름-지향 전송(흐름-指向 轉送 stream-oriented transmission)과 기록마디-지향 전송(記錄마디-指向 轉送 record-oiiented transmission)이다.

흐름-지향 전송에서, 자료 집합에 있는 자료는 문자 꼴로 된 자료 항목의 연속 흐름이라 여겨진다. 결과로서, 입력 흐름에 있는 문자들은 번역이 되고 필요할 때는 지정된 내부 꼴로 변환된다; 출력에서, 내부 꼴의 자료 항목은 필요할 때는 문자 꼴로 변환되고 출력 흐름에 추가된다. 흐름-지향 전송에 사용되는 자료 전송문은 GET와 PUT이다. 입력 자료 항목이 배치될 변수와 출력 자료 항목이 전송될 식은 보통 각개의 GET나 PUT에서 자료 나열(資料 羅列 data list)로 지정된다.

자료 집합에 있는 항목이 기록마디 형식으로 존재하지만, 흐름 전송에서는 그런 구성은 프로그램 안에서 무시되고 자료는 개개 자료 항목의 연속 흐름인 것처럼 취급된다.

기록마디-지향 전송에서, 자료 집합에 있는 자료는 계산기에 사용할 수 있는 어떤 서식(書式 format)으로도 기록된 불연속(不連続)한 논리 기록마디의 집합체로 여겨진다. 기록마디 전송중에 자료 변환이 이루어지지 않는다; 입력에서 이는 자료 집합에 기록된 대로 정확하게 전송되고, 출력에서 이는 내부에서 기록된 대로 정확하게 전송된다.

READ, REWRITE, WRITE 문이 한개의 논리 기록마디를 자료 변수로 또는 자료 변수로 부터 전송시키고, SET 선택항을 가진 READ의 경우에는 임시로 주소를 정한 완충역(緩衝域 buffer)으로 전송시킨다. LOCATE문은 기록마디를 위한 자료가 배치될 완충역에서 장소를 할당한다.

기록마디가 블러크 되었지만, 이 경우 물리 기록마디가 한꺼번에 자료 집합에 또는 자료 집합으로 부터 전송된다.

기록마디 입/출(入/出 I/O)에서 각개의 자료 전송 문은 논리 기록마디와 관계를 맺는다. 블러크 된 기록마디는 자동으로 블러크가 해체된다.

제2절 화 일(file)

원시 프로그램(原始 프로그램 souree program)이 자료 집합에서 이것의 물리적 구성보다는 주로 자료의 논리적 관점을 취급하게끔, PL/I은 화일(file)이라고 하는 자료 집합의 상징적 표현을 채용한다. 이 상징적 표현은 어떻게 입력과 출력 문이 연관된 자료 집합에 접근하고 처리할 것인가를 결정한다. 자료 집합과 달라서, 화일은 원시 프로그램 안에서만 의미를 가지고

프로그램 밖에서 물리적 실체로서 존재하지 않는다.

PL/I은 화일을 위해서 선언된 화일 이름(file name)이 있어야 하고 화일 속성(화일 属性 file attribute)이라 부르는 열쇠말을(keyword) 가지고 선언된 화일의 속성을 허용한다. 화일 속성은 화일 이름에 지정된다. 화일 속성은 한개의 화일 선언에서 인수(因数)로 될 수 있다.

1. 화일 속성

다음에 나열한 것은 각개 형식의 자료 전송에 응용할 수 있는 화일 속성을 보인다.

기록마디 전송	흐름 전송
FILE	FILE
RECORD	STREAM
INPUT	INPUT
OUTPUT	OUTPUT
UPDATE	PRINT
ENVIRONMENT	ENVIRONMENT
SEQUENTIAL	
DIRECT	
BUFFERED	
UNBUFFERED	
KEYED	
BACKWARDS	

이들 각개의 속성들에 관한 자세한 설명은 제Ⅱ부. 제9장, "속성"에 있다. 다음에 있는 논술은 각 속성의 간략한 설명을 보이고 어떻게 속성이 화일에 선언되는 가를 보인다.

〔1〕 FILE 속성

화일 속성(file attribute)은 연관된 표식어가 화일 이름임을 가리킨다. 보기로, 표식어 MASTER는 다음에서 화일 이름임이 선언되었다.

DECLARE MASTER FILE . . . ;

FILE속성은 생략될 수도 있다;이는 화일 서술 속성에 의해 암시된다. FILE이 지정된다면, 이는 화일 이름 뒤에 오는 첫 번째 속성이 꼭 될 필요는 없다.

〔2〕 택일하고 부가하는 속성(択一하고 附加하는 属性 alternative and additive attributes)

FILE속성과 연관된 속성은 두개 범주(範疇)로 갈라진다;택일하는 속성과 부가하는 속성. 택일하는 속성이란 속성의 한 종류로 부터 뽑아내는 것이다. 만일 택일 속성에 하나도 선언이 없고 택일 속성이 요구된다면, 태만 속성이 대부분 대행된다.

부가 속성이란 명시해서 적어야 하거나 다른 명시해서 적어넣은 속성에 의해서 암시되는 것이다. 부가 속성 KEYED는 DIRECT 속성에 의해 암시될 수 있다. ENVIRONMENT속성은 모든 화일에 항상 명시로 선언되어야 한다. 부가 속성은 태만에 의해 주어질 수 없다.

〔3〕 택일 속성

PL/I은 네 종류의 택일하는 화일 속성을 준비한다. 각

개 종류는 별개로 논술된다. 다음은 모임의 나열과 각각을 위한 태만이다 ;

모 임	택 일	태 만
형 식	속 성	속 성
용 법	STREAM|RECORD	STREAM
기 능	INPUT|OUTPUT|UPDATE	없 음
접근 (接近 acess)	SEQUENTIAL|DIRECT	SEQUENTIAL
완 충 역	BUFFERED|UNBUFFERED	BUFFERED

주 : 기능 속성에는 태만 속성이 사용 안된다 ; 하나가 항상 지정되어야 한다. UNBUFFERED화일의 경우, INPUT와 OUTPUT가 DECLARE 문에서가 아니고 OPEN 문에서 나타날 수 있다. 화일 이름의 범위는 언제나 EXTERNAL. 화일 이름은 이 속성을 가지고 명시 선언될 수 있다 ; 아니면 이는 자동으로 제공된다.

〔 4 〕 STREAM과 RECORD 속성

STREAM과 RECORD 속성은 이 화일을 위한 입력과 출력 연산에서 사용되는 자료 전송 형식 (흐름-지향 또는 기록마디-지향) 을 기술한다.

STREAM 속성은 화일과 연관된 자료 집합을 문자 꼴 (form) 로서만 기록된 자료 항목의 연속 흐름으로서 취급하게 한다.

RECORD 속성은 화일과 연관된 자료 집합을 논리 기록마디로 이어져 있는 것으로 취급하게 한다. 각 기록마디는 실제에서 허용하

는 어떤 내부 꼴로서도 기록된 하나 이상의 자료 항목으로 구성된다.

```
DECLARE MASTER FILE RECORD . . . ,
        DETAIL FILE STREAM . . . ;
```

〔 5 〕 INPUT, OUTPUT, UPDATE 속성

기능 속성은 화일에 허용되는 자료 전송의 방향(方向)을 결정한다. INPUT는 읽는데, OUTPUT는 쓰는데, UPDATE는 읽고 쓰는 양편으로 되는 화일에 제공된다.

```
DECLARE  DETAIL  FILE INPUT . . . ,
         REPORT  FILE OUTPUT . . . ,
         MASTER  FILE UPDATE . . . ;
```

〔 6 〕 SEQUENTIAL과 DIRECT 속성

접근 속성은 기록마디 속성을 가진 화일에만 제공되고 화일에 있는 기록마디가 어떻게 접근되는 가를 기술한다.

SEQUENTIAL 속성은 연속한 기록마디가 자기 테이프처럼 그들의 연속한 물리적 위치에 기초해서 접근되는 것을 지정한다.

DIRECT 속성은 화일에 있는 기록마디가 화일에 있는 기록마디의 자리에 기초해서 접근되고 앞에서 읽혔거나 쓰여진 기록마디에 대해 상대적 위치에 기초하지 않는다. 기록마디의 자리는 열쇠(key)에 의해 결정된다 ; 그러므로, DIRECT 속성은 KEYED 속성을 암시한다. 연관된 자료 집합은 직접-접근 장치에 있어야 한다.

〔 7 〕 BUFFERED와 UNBUFFERED 속성

완충역을 만드는 속성은 연속적 SEQUENTIAL과 RECORD 속성을 가진 화일에 제공된다. BUFFERED 속성은 또한 INDEXED SEQUENTIAL 화일에 지정될 수 있다. BUFFERED 속성은 전송될 기록마디가 임시로 만든 내부 기억역을 통해서 지나가야 함을 가르킨다. 완충역의 크기(size)는 보통 블러크의 크기와 같다. 이는 자동으로 기록마디를 블러크로 만들고 블러크를 풀기도 한다.

UNBUFFERED 속성은 논리 기록마디가 완충역을 지나지 않고 직접 전송됨을 가르킨다. 논리 기록마디와 물리 기록마디는 UNBUFFERED 화일을 가진 자료 집합에서 같은 크기이다.

주 : D-편성자에서. UNBUFFERED 속성은 언제나 기록마디가 어떤 완충역이나 임시 기억역을 지나지 않음을 지정한다. 이른바 "숨은 완충역"은 사용 안된다.

UNBUFFERED는 카드, 인쇄기, 2321 화일에는 허용 안된다.

〔 8 〕 부가 속성(additive attributes)

부가 속성은 :

PRINT, BACKWARDS, KEYED, ENVIRONMENT (선택항 나열)

〔 9 〕 PRINT 속성

PRINT 속성은 STREAM과 OUTPUT 속성을 가진 화일에만 제공된다. 이는 이것이 다른 어떤 매체에 쓰였더라도 마지막에는 인쇄될 것임을 가르친다. PRINT 속성은 연관된 기록마디가 인쇄 통제 문자로서 확보된 최초의 글자를 가지고 생성됨을 지정한다.

〔10〕 BACKWARDS 속성

BACKWARDS 속성은 화일이 마지막 논리 기록마디에서 시작해서 처음 기록마디로 처리되는 역순(逆順)으로 접근됨을 가리킨다. BACKWARDS 속성은 자기 테이프에 자료 집합이 있으며 SEQUENTIAL과 INPUT 속성을 가진 RECORD 화일에만 제공된다.

〔11〕 KEYED 속성

KEYED 속성은 화일의 각 기록마디는 열쇠(key)를 가지며 자료 전송 문의 열쇠 선택항(KEY나 KEYFROM)을 사용해서 접근될 수 있음을 가리킨다. KEYED 속성은 실제의 열쇠가 존재하거나 자료 집합에서 기록되었음을 가리키는 것이 아니라는 걸 유의하라. STREAM과 PRINT 속성은 KEYED 속성을 가진 화일에는 제공되지 못한다. 열쇠의 사용은 이 장에 있는 "자료 집합을 위한 여러가지 고려점"과 "기록마디-지향 전송"에서 상세히 논술된다.

〔12〕 ENVIRONMENT 속성

ENVIRONMENT 속성은 화일과 연관된 자료 집합의 물리적 구성에 대한 정보를 지정한다. 이런 특성은 ENVIRONMENT 속성 지정에서 괄호에 싸인 선택항 나열에서 지정된다. D-편성자에서 선택항 나열은 "자료 집합을 위한 여러가지 고려점"에서 논술된다

주 : 이장 앞에서 논의된 것처럼, 각 화일은 명시 선언되어야 한다 ; ENVIROMENT 속성은 모든 화일에 나타나야 한다.

2. 화일을 열고 닫기 (opening and dosing files)

화일과 연관된 자료가 입력이나 출력 문에 의해 전송되기 전에

어떤 화일 준비 활동이 일어나야 한다. 이는 외부 기억 매체의 사용 가능성 검토, 매체를 제자리에 갖다놓기, 적당한 프로그램 지원을 할당하기 등이다. 이런 활동은 화일을 여는 것으로 알려진다. 또한 처리가 끝났을 때, 화일은 닫쳐야 한다. 화일을 닫는 것은 화일을 여는 동안에 설정되었던 시설을 철수하는 것을 포함한다.

PL/I 서브세트(PL/I subset)는 이들 기능을 수행하는 두개의 문 OPEN과 CLOSE를 준비한다. RECORD 속성을 가진 모든 화일은 사용되기 전에 명시로 열려야 한다. 그렇지만, STREAM 화일에서, 명시 열기는 선택적이다. 만약 OPEN 문이 STREAM 화일을 위하여 실행되지 않으면, 이 화일은 첫번째 GET나 PUT 문이 실행될 때 자동으로 열린다; 이 경우, 자동적 화일 준비는 명시 OPEN이 GET나 PUT 문 전에 실행된 것과 같다. 모든 화일은, STREAM과 RECORD, 프로그램이 끝나기 전에 닫치지 않았어도 프로그램이 끝남에 따라 자동으로 닫힌다.

두가지를 제외하고는, 모든 화일은 다시 열릴 수 있다. 다시 여는 것은 〔1〕 INDEXED 화일과 〔2〕 SYSIPT, SYSPH, SYSLST가 MEDIUM 선택항에서 지정된 화일에는 허용 안된다.

테이프 타래(tape reels)는 LEAVE 선택항이 ENVIRONMENT 속성에서 지정되지 않는 한 테이프 화일을 열고 닫을 때 자동으로 감긴다.

〔1〕 OPEN 문

한개의 OPEN 문의 실행은 명시로 하나 이상의 화일을 열 수 있다. OPEN 문은 다음과 같은 기본 형식을 가진다.

OPEN FILE (화일 - 이름) 〔 선택항 - 나열 〕

〔. FILE (화일 - 이름) 〔 선택항 - 나열 〕〕 . . . ;

선택항 나열 (option list) 은 UNBUFFERED 속성을 가진 화일에 준비되는 INPUT나 OUTPUT가 될 수 있다. 여기는 PAGESIZE 선택항이 될 수도 있다.

〔 2 〕 암시 열기 (implicit opening)

암시 열기는 어느 화일에 대해 앞에서 OPEN 문의 실행 없이 GET나 PUT문이 실행될 때만 일어난다. 암시 열기의 효과는 OPEN과 같다. GET 문으로서 암시로 열리는 모든 화일은 INPUT로서 명시 선언되어야 하고, PUT 문으로 암시로 열리는 모든 화일은 OUTPUT로서 명시 선언되어야 한다.

〔 3 〕 속성의 혼합

화일 선언에서 지정된 속성과 명시 화일 열기의 결과로서 혼합된 속성간에 모순이 없어야 한다. 보기로, 화일이 DECLARE 문에 BACKWARDS 속성을 받고 OPEN 문에서 OUTPUT 문을 받을 때 모순이 존재한다. 속성 BACKWARDS와 OUTPUT는 모순 관계에 있으므로, 오류 전언문이 프로그램 편성시에 생성된다.

〔 4 〕 화일과 자료 집합의 연관

D - 편성자에서, 화일 이름은 PL/I 서브세트 ENVIRONMENT 속성에 있는 MEDIUM 선택항을 이용해서 자료 집합과 연관되고, 필요하다면, DOS/TOS J'ob Control Language의 ASSGN 문을 사용한다. 자료 집합과 화일을 연관시키는 방법은 이 장 뒤에 있는 " ENVIRONMENT 속성 " 제목 아래에 있는 MEDIUM 선택항의 논술에서 기술된다.

〔 5 〕 CLOSE 문

CLOSE 문의 기본 형식은:

CLOSE FILE (화일 - 이름) 〔, FILE (화일 - 이름) 〕 . . . ;

CLOSE 문의 실행은 자료 집합으로 부터 지정된 화일을 분리시킨다. CLOSE 문은 또 화일로 부터 명시 열기로 설정된 INPUT 나 OUTPUT 속성을 분리한다. 원한다면, 새로운 INPUT 나 OUTPUT 속성이 다음에 오는 OPEN 문에서 지정될 수 있다.

주: 이미 닫힌 화일을 닫거나 이미 열린 화일을 여는 것은 효과가 없다.

3. 인쇄 화일의 지면배정 (紙面配定)

PRINT 속성을 가진 화일에서 지면의 총체적 양식은 OPEN 문의 PAGESIZE 선택항의 방법으로 관리된다.

보기 :

```
DECLARE  REPORT FILE OUTPUT PRINT
         ENVIRONMENT ( 선택항 - 나열 ) ;
OPEN  FILE ( REPORT )  PAGESIZE ( 55 ) ;
```

지정 PAGESIZE (55) 는 각 지면은 최대 55 줄로 될 수 있음을 가리킨다. 55 줄을 지난뒤에 쓰려고 시도하면 ENDPAGE 조건이 일어날 것이다. PAGESIZE 조건에 대한 표준 조직 행위는 새로운 지면으로 보내는 것이다. 그러나 프로그램에는 ON 문을 사용해서 그 자신의 행위를 설정할 수 있다.

ENDPAGE 조건은 매 지면에 한번만 일어난다. 이 결과로, 인쇄가 지정된 줄을 넘어서 계속될 수 있다.

다음은 지면의 아래에 설명을 쓰는 경우이다.

```
ON ENDPAGE ( REPORT )  GO TO FOOT ;
. . .
FOOT : PUT FILE ( REPORT )  SKIP EDIT ( FOOTING ) (A) ;
       PUT FILE ( REPORT )  PAGE ;
       N = N + I ;
       PUT FILE ( REPORT )  EDIT ( 'PAGE', N ) ( A, F(3) ) ;
       PUT FILE ( REPORT )  SKIP (3) ;
       GO  TO  NEXT ;
```

한 줄에 인쇄될 수 있는 문자의 최대수(이는 줄 크기)는 ENVIRONMENT에서 지정된 고정 길이 기록마디 크기와 같다. 새 줄이나 지면으로 건너가지 않고 최대의 지정된 문자 수보다 더 많이 쓰려고 한다면 남는 문자는 다음 줄에 쓰여진다. PAGESIZE 선택항은 PRINT 속성을 가진 화일에만 지정될 수 있고 이는 OPEN 문에서만 지정될 수 있다.

PRINT 화일에서 기록하는데 관한 더 자세한 것은 이 장에 있는 "자료 전송"을 보라.

4. 표준 화일 (標準 화일 standard file)

두개의 표준 조직 화일이 (標準 組織 화일 standard system files) 어떤 PL/I 서브세트 프로그램에서 사용될 수 있도록 준비되어 있다. 이들 화일은 PL/I 서브세트에서 FILE도 STRING도 없는 GET나 PUT 분을 지정하므로써 인용된다.

보기 :

GET EDIT . . . ;　　PUT EDIT . . . ;

위의 GET에서, DOS/TOS 조직 입력 장치 SYSIPT가 인용된다 ; PUT에서, DOS/TOS 출력장치 SYSLST가 인용된다. 이들 표준 입력/출력 장치가 위에서와 같이 인용된다면, 명시 화일 선언이 없어도 된다. 그렇지만, ENDFILE이나 ENDPAGE 조건이 일어나지 않는다.

제3절 자료 집합을 위한 여러가지 고려점

D-편성자에 의해 생산되어 편성된 PL/I 프로그램은 DOS/TOS 통제 아래서 실행되도록 설계되었다. 이는 자료 집합의 구성, 위치, 기억소, 회수를 통제하는 자료 관리 설비(프로그램)를 준비한다 PL/I은 이것이 실행될 때 이런 설비를 불러낸다.

1. 입력과 출력의 기계 독립성

PL/I 서브세트 프로그램의 입력과 출력 문은 자료 집합의 논리적 구성에 관계되고 이것의 물리적 특성에 관계하지 않는다. 입력/출력 장치 번호와 기록 밀도(記錄 密度 recording density)와 같은 정보는 PL/I 프로그램이 실행될 준비가 되기까지는 필요없다. 입력/출력 장치 형식과 완충역을 만드는 방법과 같은 정보는 ENVIRONMENT 속성에서 분리된다. 이런 장치 독립성(device independence)은 PL/I 프로그램을 바꾸지 않거나 ENVIRONMENT 속성만 바꾸므로써 이런 정보를 바꾸는 것이 가능하다. 특정 입/출 장치에 대해 요구되는 정보는 ENVIRONMENT 속성의 MEDIUM을 통해 공급된다.

장치 형식은 MEDIUM에서 SYSLST, SYSPH, SYSIPT가 사용될 수 있다.

〔 1 〕 ENVIRONMENT 속성

ENVIRONMENT 속성은 화일과 연관된 자료 집합의 물리적 구성에 관한 정보를 준비한다. 이 정보는 편성자가 자료 집합에 접근하는 방법을 결정하게 한다.

D-편성자에서, ENVIRONMENT 속성은 다음의 일반적 형을 가진다.

```
ENVIORONMENT  [ CONSECUTIVE                  ]
              [ REGIONAL ( { 1 | 3 } )       ]
              [ INDEXED                      ]

  [ F ( 블러크크기 〔 , 기록마디 크기 〕 )  ]
  [ V ( 최대 블러크크기 )                  ]
  [ U ( 최대 블러크크기 )                  ]

〔 BUFFERS ( { 1 | 2 } )〕
  MEDIUM ( 논리-장치-이름, 물리-장치-형식 )
〔 CTLASA|CTL360 〕〔 LEAVE 〕〔 NOTAPEMARK 〕〔 NOLABEL 〕
〔 VERIFY 〕〔 KEYLENGTH ( 십진-고른수-정수 ) 〕
〔 EXTENTNUMBER ( 십진-고른수-정수 ) 〕
〔 INDEXMULTIPLE 〕〔 HIGHINDEX ( { 2311|2314|2321 } ) 〕
〔 OFLTRACKS ( 십진-고른수-상수 ) 〕
〔 INDEXAREA ( 십진-고른수-상수 ) 〕
〔 ADDBUFF ( 십진-고른수-상수 ) 〕 )
```

논술의 편의를 위해, ENVIRONMENT 속성의 선택항은 네 부분으로 나뉜다 : 기록마디 서식 (record format), 자료 집합 구성

(data set organization). 장치 할당 (device allocation), 자료 집합과 연관된 열쇠의 길이 (length of key), 기타. 기록마디 형식은 V, F, U중에 하나가 지정되어야 한다. 자료 집합 구성의 하나가 지정되어도 된다 (지정 없으면 CONSECUTIVE가 태만에 의해 제공된다). MEDIUM 지정은 항상 나타나야 한다. 그 외의 선택항은 자료 집합의 모양과 용도에 따라 나타나거나 나타나지 않거나 한다. 완전한 화일 선언의 보기는 제13장, "PL/I 프로그램"에서 찾을 수 있다.

〔2〕 기록마디 형식 (記錄마디 形式 record format)

논리 기록마디는 세가지 형식 중에 하나로 나타난다: 고정 길이 (F-형식) (fixed length (F-format), 가변 길이 (V-형식) (variable length) (v-format), 부정 길이 (U-형식) (undefined length) (U-format).

블러크 크기와 기록마디 크기는 글자 수로 지정된다. F-형식에서, 만일 기록마디 크기가 지정되지 않으면, 기록마디는 블러크 안된 것으로 대행된다. 블러크 크기는 꼭 지정되어야 한다.

기록마디 크기는 F-형식 기록마디에만 지정될 수 있다. 블러를 만들고 해체하는 것은 자동으로 처리된다. F-형식 기록마디에서, 블러크를 해체하는 것은 쓰여진 기록마디 크기에 의존한다 블러크 크기는 기록마디 크기의 고른 수 배 (倍)이어야 한다.

V-형식 기록마디는, 블러크를 해체하는 것은 각 블러크의 처음과 각 기록마디의 처음에 있는 정보에 의존한다. 4글자가 블러크 길이를 지정하기 위하여 각 블러크의 처음에 사용되고, 4글자가 기록마디 길이를 지정하기 위하여 각 기록마디의 처음에 사용된다. 이런 길이 정보를 삽입하는 것은 자료 집합이 만들어

질 때 조직에 의해 자동으로 이루어지지만, 프로그램에는 이 길이 지정 글자 수를 포함해야 한다. V-형식 자료 집합이 만들어질 때, 기록마디는 둘 이상의 기록마디가 지정된 최대 수보다 작거나 같은 길이에 들어가면 언제나 블러크 된다.

U-형식 기록마디에서, 각 블러크는 한개의 기록마디만으로 된다. 블러크(기록마디)는 길이가 변한다. 조직 통제 글자는 블러크 안에 없다. 기록마디에 관한 모든 처리는 프로그래머의 책임이다.

〔3〕 자료 집합 구성(data set organization)

PL/I은 세가지의 자료 집합 구성이 있다: CONSECUTIVE, REGIONAL, INDEXED. CONSECUTIVE 구성은 태만에 의해 대행된다.

CONSECUTIVE 자료 집합 구성: CONSECUTIVE 구성을 가진 자료 집합에서, 논리 기록마디는 순전히 연속되는 물리적 위치를 기초로 구성된다. 기록마디는 순차로만 읽어내진다. 그러므로 연관된 화일은 SEQUENTIAL 속성(또는 STREAM 화일)을 가져야 한다. 기록마디는 F-형식, V-형식, U-형식이 될 수 있다. 마지막 두 형식(V와U)은 RECORD 입력/출력과 테이프와 직접 접근 장치에만 사용될 수 있다.

CONSECUTIVE 자료 집합에 허용되는 입력/출력 장치는 자기 테이프, 카드 독취기와 천공기, 직접 접근 기억 장치, 인쇄기 등이다.

CONSECUTIVE 자료 집합이 만들어진 뒤에, 이는 INPUT 또는 UPDATE로서만 열린다. 이런 자료 집합을 읽는 것은 만일 자료 집합이 자기 테이프에 기록되어 있으면 앞으로나 뒤로나 될 수 있

다.

ENVIRONMENT 속성의 CONSECUTIVE 선택항과 SEQUENTIAL 속성간에는 차이가 있음을 유의하라. CONSECUTIVE는 자료 집합의 물리적 구성을 지정하고; SEQUENTIAL은 어떻게 화일이 처리되는가를 지정한다. 그렇지만, PL/I 서브세트에서, CONSECUTIVE 구성을 가진 자료 집합은 SEQUENTIAL 화일과 연관되어야 하고, REGIONAL 구성을 가진 자료 집합은 DIRECT 화일과 연관되어야 하고, 반면 INDEXED 구성 자료 집합은 DIRECT나 SEQUENTIAL 화일과 연관될 수 있다.

REGIONAL 자료 집합 구성 : 자료 집합의 REGIONAL 구성은 자료 집합에 있는 기록마디의 물리적 자리잡기의 통제를 제공한다. 이런 종류의 통제는 프로그래머에게 특정한 기록마디에 접근하는 시간을 최소로 해준다. REGIONAL 자료 집합이 허용되는 입력/출력 장치는 직접 접근 기억 장치에 한한다.

기록마디 열쇠(record key) : REGIONAL 자료 집합은 특정 기록마디를 찾기 위해 열쇠의 사용을 허용한다. 두 종류의 열쇠가 있다. 기록된 열쇠(recored key)와 원시 열쇠(原始 열쇠 source key). 기록된 열쇠란 기록마디의 증명으로서 매 기록마다 자료 집합에 나타나는 문자 줄이다. 이는 255글자를 넘지 못한다. 원시 열쇠란 기록마디 지향 자료 전송에서 문이 인용하려는 기록마디를 찾으려고 나타나는 문자 줄(또는 요소 식)이다.

열쇠가 지정되고 사용되는 방법은 PEGIONAL 구성의 두가지 종류간에 다르다. 원시 열쇠가 프로그램에서 자료 집합에 접근하거나 자료 집합을 만들 때(KEY 또는 KEYFROM 선택항을 사용해서), KEYED 속성이 그 화일에 지정되어야 한다. 덧붙여, 기록된 열쇠

를 포함하는 자료 집합에서, ENVIRONMENT 속성의 KEYLENGTH 선택항이 글자 수로 기록된 열쇠의 실제 길이를 지정하는데 사용되어야 한다.

두 종류의 지역 지정이 사용된다. 상대 기록마디와 상대 궤도 (relative record and relative track). 상대 기록마디 지정은 자료 집합에서 번호 영을 가지는 첫번째 기록마디에 대해 상대적인 특정 기록마디의 번호를 지정하므로써 자료 집합의 지역을 인용한다. 상대 궤도 지정은 자료 집합에서 번호 영을 가지는 첫번째 궤도에 대해 상대적인 특정 기록마디의 번호를 지정하므로써 자료 집합의 지역을 인용한다. 상대 궤도나 상대 기록마디 지정은 항상 자료 집합에서 한개의 지역을 인용한다.

REGIONAL 자료 집합의 순차 처리는 허용되지 않는다. 모든 REGIONAL 자료 집합은 DIRECT 속성을 가진 화일과 연관되어야 한다. 다음 REGIONAL의 각 형식에 대해 논술한다.

REGIONAL (1) 구성 (構成 organization) :

REGIONAL (1) 구성을 가진 자료 집합은 기록된 열쇠 (recorded key) 를 갖지 않은 블러크 안된 (unblocked) F-형식 기록마디로 이루어진다. 자료 집합에 있는 각 지역은 하나의 논리 기록마디로만 이루어진다 ; 고로 각 지역 번호 (地域 番号 regianal number) 는 자료 집합 안에서 한개 논리 기록마디의 위치 (位置 position) 를 표시한다. 첫번째 기록마다의 상대 위치는 영이다. 여기에는 기록된 열쇠가 없으므로, 지역 번호만이 원시 열쇠로서 의미를 가진다. 원시 열쇠의 문자 줄 값은 16777215를 넘지 않는 부호없는 십진 고른수를 표현해야 한다. 문자 0부터 9까지만이 사용된다 (앞서는 빈자는 0으로 해석되지 않는다) 어떤 원시

열쇠 식도 수치 문자 0부터 9까지 만으로 되어 있는 길이 8의 문자 줄 결과로 되어야 한다. 이를 행하는 한가지 방법은 PICTURE'(8)9' 속성을 사용해서 수치 문자 변수로서 원시 열쇠를 선언하는 것이다.

REGIONAL(3) 구성 : REGIONAL(3) 구성을 가진 자료 집합은 기록된 열쇠를 가지는 블러크 안된 F-형식 기록마디로 이루어진다. REGIONAL(1) 구성과 달라서, 자료 집합의 각 지역은 직접-접근 기억 장치의 궤도와 일치한다. 그러므로 둘 이상의 논리 기록마디를 포함할 수 있다.

각 논리 기록마디와 연관된 기록된 열쇠는 자료 집합에 기록되었으며 기록마디 바로 앞에 있는 문자 줄이다. 기록된 열쇠는 항상 가장 오른쪽 8문자로서 지역 번호를 가져야 한다. 원시 열쇠는(상수나 다른 식으로서 지정된다) 문자-줄 값이 된다. 이는 두개의 논리적 부분을 가지는 것으로 생각된다. 지역 지정과(region specification) 지역내에서 그 기록마디를 고유하게 증명하는 문자줄 열쇠의 지정이다. 원시 열쇠 식(또는 단일 변수)은 길이가 ENVIRONMENT 속성의 KEYLENGTH 지정과 같은 문자 줄이 되어야 한다. 원시 열쇠의 가장 오른쪽 8개 문자는 지역 지정(region specification)을 이루며, 이는 지역 번호를 말한다. (문자 0부터 9까지, 빈자는 0으로 해석되지 않는다) 이 8문자를 제외한 것은 문자-줄 열쇠 지정(character-string key specification)이 된다. 기록마디를 읽어내려면, 원시 열쇠와 기록된 열쇠가 꼭 맞아야 한다. 열쇠 길이는 적어도 9자 이상이어야 한다

보기 : 원시 열쇠

KEY ('JOHNbDOEbbb00003251')

오른쪽 8자 00003251은 궤도의 상대 번호이다. 여기서 이 전체길이와 같은 기록된 열쇠를가지며 내용이 같은 기록마디를 찾는다. 만일 기록마디가 3251에서 발견되지 않으면 KEY조건이 일어난다.

만일 위 열쇠가 출력 연산에서 사용된다면. 기록마디는 궤도 3251에서 첫번째 사용하가능한 자리에 기록될 것이다. 만일 빈 자리가 없으면. KEY조건이 일어난다.

REGIONAL(3)을 위한 지역 지정은 16777215를 넘을 수 없다.

REGIONAL 형식의 비교 : REGIONAL 자료 집합에 있는 기록마디는 유효한 값을 표시하는 "실"("実" "actual")또는 자료 집합이 만들어질 때 준비된 사용 가능한 영역을 표시하는 "가"("仮" "dummy") 기록마디이다. F-형식 기록마디만 REGIONAL에 허용된다.

외
6

화일이 REGIONAL 자료 집합을 만들기 위해 열릴 수 있기에 앞서. 사용될 화일의 모든 범위는 DOS Clear Disk편의 프로그램을 사용해서 준비를 해놓아야 한다. 이 프로그램은 가 기록마디(즉 임시 기록마디)를 만든다. 이 기록마디는 사용자가 정의한 문자로 된 문자 줄로 내용을 이루고 모든 궤도가 비게끔 용량 기록마디 RØ를 다시 놓는다. (RØ의 Ø는 영을 나타냄)

REGIONAL(3)에서. RØ를 다시 놓는것은 가 기록마디는 실 자료 기록마디로서 회수 안되는 것을 보장한다. REGIONAL(3)화일에서, 모든 가 기록마디는 첫번째 실 기록마디가 한개의 궤도에 기록된 뒤에는 그 궤도로 부터 제거된다. (자세한 것은, IBM조직/360 원반과 테이프 운영 조직, 편의 프로그램 설명서. 주문 번호

GC24-3465. (IBM System/360 Disk and Tape Operating Systems, Utility Program Specifications, Order NO. GC24-3465.)를 보라). 한번 그 권(卷 Volume)의 형식이 이 편의 프로그램에 의해 설정되면, 화일이 열릴 수 있고 REGIONAL자료 집합이 생성될 수 있다. 화일은 물론 DIRECT 속성을 가져야 한다. 이는 SEQUENTIAL은 REGIONAL 자료 집합에 허용되지 않기 때문이다.

기록마디를 읽어내려면, REGIONAL 자료 집합과 연관된 화일은 INPUT나 UPDATE 속성을 가질 수 있다. 이는 DIRECT 속성을 가져야 한다.

REGIONAL 자료 집합이 UPDATE 속성을 가진 화일과 연관된다면, 기록마디는 회수되고(읽어냄), 추가되고, 바꾸어 놓아질수 있다.

① 회수(回收 retrieval)

REGIONAL(1) : 가 또는 실의 모든 기록마디가 회수될 수 있다.

REGIONAL(3) : 가 기록마디는 회수되지 못함.

② 추가(追加 addition)

REGIONAL(1) : 추가는 가 또는 실이건 모든 기록마디의 교환을 포함한다. (오류 조건은 어느 경우건 일어나지 않음).

REGIONAL(3) : 추가는 지정된 기록마디 속으로 기록마디를 위치시키는 것을 포함한다.

③ 교환(交換 replacement)

REGIONAL(1) : 가 또는 실이건 지정된 기록마디는 바꿔쓸 수

있다.

REGIONAL(3) : 지정된 열쇠를 가진 기록마디가 존재해야 한다 이 기록마디는 바꿔 써진다.

INDEXED 자료 집합 구성 : INDEXED 구성을 가진 자료 집합은 이는 직접 접근 장치에 있어야 한다. 모든 기록마디와 연관된 열쇠에 따라 논리적 순서로 배열된 기록마디를 가지고 있다. 열쇠는 보통 기록마디 안에서 한개의 항목을 표시하는 문자 줄이다. 이는 부분. 번호, 날자. 이름 같은 것이 될 수 있다. 논리 기록마디는 대조 순서에 따라서. 이들 열쇠의 오르막 순으로 배열된다. 각 궤도와 각 원통(Cylinder 円筒)에서 가장 높은 열쇠를 지정하는 찾아보기 (索引 indexes) 가 있다.

블러크 되거나 안된 F-형식만이 INDEXED 구성에 사용될 수 있다.

기록된 열쇠의 길이는 ENVIRONMENT에서 선택항 KEYLENGTH에서 지정되어야 한다. 기록마디에 있는 자료의 일부로 포함된 열쇠가 ENVIRONMENT에서 선택항 KEYLOC(n)에 지정되어야 한다. 여기서 n은 기록마디에서 열쇠의 가장 왼쪽 자리를 나타내는 십진 고른수 상수이다. 보기로 열쇠가 기록마디의 두번째에서 시작되면, KEYLOC(2)라 지정되어야 한다. INDEXED 자료 집합을 만드는 과정에서 WRITE 문에서 KEYFROM 선택항에서 유래하는 열쇠는 KEYLOC가 지정되었다면 자동적으로 옮겨진다.

주 : 블러크 안된 기록마디에서 열쇠는 항상 실 자료 앞에 있는 자리에 기록된다. 그런 이유로, KEYLOC는 이 경우 지정될 필요가 없다. 그렇지만, 만일 이것이 지정되면, 블러크 안된 기록마디는 블러크된 기록마디와 같은 방법으로 처리된다.

CONSECUTIVE 구성과 같지 않아서, INDEXED 구성은 모든 기록마디를 순차 방식으로 접근되는 것을 요구하지 않는다. 임의 회수 (任意 回收 random retrieval) . 추가 . 교환이 허용된다. 그렇지만, 만일 프로그램이 원한다면, 순차 접근 (順次 接近 sequential acess) 이나 직접 접근이 사용될 수 있다.

INDEXED에서 열쇠는 최대 255문자의 문자 줄이다. 이는 항상 자료 집합 안에 기록되며 보통으로 자료의 일부이다. INDEXED 자료 집합에 있는 기록마디는 기록마디-지향 전송 문을 사용해서 접근된다. 만일 접근이 직접이면, 이들 문은 기록된 열쇠의 값에 의해 특정 기록마디와 일치하는 원시 열쇠를 제공한다.

비교 :

INDEXED 자료 집합은 순차로만 생성될 수 있다. 한번 INDEXED 자료 집합이 생성된 뒤에는, 이와 연관된 화일은 INPUT나UPDATE 속성이면 SEQUENTIAL이나 DIRECT 속성을 가져야 한다. 화일이 DIRECT 속성을 가지면, 기록마디는 임의로 회수되고, 추가되고, 교환될 수 있다.

F-수준 편성자에서, INDEXED 자료 집합에 있는 논리 기록마디는 만일 자료의 첫번째 글자가 상수 (8) '1'B로 되어 있으면 가 기록마디이다. 자세한 것은 PL/I 참고 지침서, Form C28-8201 (PL/I Reference Manual, Form C28-8201) 를 참조하라.

SEQUENTIAL INDEXED 화일 : SEQUENTIAL 방식으로 접근되는 INDEXED 자료 집합의 화일은 INPUT나 UPDATE 속성으로 열릴 수 있다. 그러나 자료 전송 문은 원시 열쇠를 포함할 필요는 없다. 순차 접근은 오르막으로 기록된 열쇠 값의 순서이다. 논리 기록마디는 이 순서이며 그들이 자료 집합에 추가되는 순서일

필요는 없다.

INDEXED 자료 집합이 INPUT나 UPDATE활동에서 순차로 접근될때, 이는 READ문에 있는 원시 열쇠를 사용해서 자료 집합을 원하는 자리에 갖다 놓는 것이 가능하다. 원하는 자리에 갖다 놓는 것은 읽어낼 지정된 기록마디에 대해 뒤쪽으로 부터나 앞쪽으로 부터 일어날 수 있다. 만일 그 다음의 READ문이 원시 열쇠를 갖고 있지 않으면, 바로 다음으로 높은 기록된 열쇠를 가진 기록마디가 회수될 것이다.

INDEXED 자료 집합의 화일이 SEQUENTIAL과 UPDATE 속성을 가지면, READ와 REWRITE만이 사용될 수 있다. REWRITE 문이 실행되기 전에, 지정된 기록마디가 READ 문에 의해서 회수되었어야 한다. 그렇지만, 회수된 모든 기록마디가 바꿔 써질 필요는 없다. 논리 기록마디는 SEQUENTIAL UPDATE 방식으로 접근되는 INDEXED 자료 집합에 추가되지 못한다.

주 : INDEXED 기록마디는 직접 기초된 변수와 함께 사용되지 못한다; 이는 SET를 가진 LOCATE와 READ는 INDEXED 화일에 허용되지 않는다.

DIRECT INDEXED 화일 : DIRECT 방식으로 접근되는 INDEXED 자료 집합의 화일은 INPUT나 UPDATE 속성을 가지고서 열릴 수 있다. DIRECT와 UPDATE 속성을 가진 화일에서, 논리 기록마디는 INTO 선택항을 가진 READ 문에 의해 회수되거나 다음 약속에 따라 추가되거나 교환된다.

① 추가(追加 addition) : 만일 열쇠가 고유하면, 논리 기록마디는 WRITE 문에 의해서 자료 집합에 삽입된다. 만약 추가되는 기록마디를 위한 빈자리가 없으면, KEY 조건이 일어난다. KEY 조건은 또 만일 추가되는 기록마디가 이미

자료 집합에 있는 기록마디의 기록된 열쇠와 같은 열쇠를 가지고 있으면 일어난다.

② 교환 (交換 replacement) : 원시 열쇠에 의해 지정된 기록마디는 REWRITE 문에 있는 새로운 기록마디로 교환된다. 기록마디는 읽지 않고 교환될 수 있다. 만일 지정된 원시 KEY가 발견되지 않으면, 열쇠 조건이 일어난다.

주 : 선택항 KEYLENGTH에서 지정된 길이는 항상 기록된 열쇠의 길이를 제공하고 원시 열쇠의 길이는 필요가 없다. 만약 원시와 기록된 열쇠의 길이가 INDEXED자료 집합이 접근될때 틀리면 이 길이들은 오른쪽에서 원저 열쇠를 자르거나 오른쪽에서 원시 열쇠가 빈문자로 늘어나거나 해서 같게 된다.

원반 운영 조직 (Disk Operating System)은 기록된 열쇠 (recorded key)를 논리 기록마디로 부터 분리하거나 논리 기록마디 안에 집어넣거나 한다.

블러크 안된 기록마디는 언제나 각 논리 기록마디 앞에 분리되어 기록된 열쇠를 가지며, 이는 열쇠가 기록마디 안에 들어 있거나 안 있거나 간에 같다. 블러크된 기록마디에서 마지막 논리 기록마디의 열쇠만이 블러크 앞에 분리되어 기록 된다.

〔 4 〕장치 할당 (装置 割当)

ENVIRONMENT 속성의 MEDIUM 선택항과, 그리고 필요하면 DOS/TOS Job Control 언어의 ASSGN 문이 화일 이름과 자료 집합을 연관시켜 주는데 사용된다. MEDIUM 선택항의 형식은 다음과 같다.

MEDIUM (논리 - 장치 - 이름, 물리 - 장치 - 형식)

논리 장치 이름(logical deaice name)이란 조직에 알려진 화일과 연관된 이름이다. 물리 장치 형식은(物理 裝置 形式 physical device type) 화일이 요구하는 입력/출력 장치의 형식을 정한다.

논리 장치 이름은 SYSXXX의 꼴(form)을 갖는다. 여기서 XXX는 IPT (조직 입력 장치), LST (인쇄에 사용되는 조직 출력 장치), PCH (카드 천공에 사용되는 조직 출력 장치), 또는 000 부터 222 까지가 될 수 있다. (000 ~ 222는 프로그래머가 정의한 논리장치) 물리 장치 형식은 사용되는 입력/출력 장치의 장치 번호를 제공하는 4자리 수이다. 보기로, IBM 자기 테이프 단위기는 2400이다 ; IBM 2311 원반 단위기에는, 2311이다.

논리 장치 이름은 프로그램이 실행되기전에 조직에 대해 가능한 특정 물리 입력/출력 단위기(単位機) (Physical input/output unit)에 배당된다. 이 배당은 두가지 방법중에 하나로 목적을 달성할 수 있다. 어떤 표준 논리 장치 이름은 자동으로 주어진 DOS/TOS 조직 배열에 있는 특정 물리 입력/출력 단위기에 연관된다.

보기 :

```
DCL  MASTER  FILE  RECORD  INPUT  SEQUENTIAL
        ENVIRONMENT ( . . .  MEDIUM ( SYSØØ6,2400 ) . . . ) ;
```

위에 대한 ASSGN 문은 다음과 같다.

```
// ASSGN SYSØØ6
```

(자세한 것은 IBM 조직/360 Disk and Tape Operating Systems. PL/I Propramer's Guide, Order NO. GC24-9ØØ5)를 참조하라.

〔 5 〕 열쇠의 길이 (length of key)

열쇠는 REGIONAL 또는 INDEXED DIRECT 또는 INDEXED SEQUENTIAL 화일에서 READ, WRITE, REWRITE에서 지정된다.

REGIONAL(1)에서, 열쇠는 지역 번호 (region number)을 지정한다. 이는 자료 집합에서 접근될 기록마디의 논리 기록마디 번호이다. 그래서, 열쇠는 단순히 8자리 수이다. 이는 문자-줄 꼴로서 논리 기록마디를 가리킨다. REGIONAL(1) 자료 집합의 열쇠의 길이는 항상 8로 된다. KEYLENGTH 선택항은 아무때고 지정되지 않는다.

REGIONAL(3) 자료 집합에서, 문자-줄 꼴인 열쇠는 기록마디가 배치되어 있는 지역을 가리키는 8자리 수와 이 지역 안에서 기록마디를 고유하게 가리키는 문자 줄이 앞에 오는 것을 지정한다.

REGIONAL(3) 화일을 위한 열쇠의 길이는 KEYLENGH 선택항을 사용해서 지정되어야 하고 8 더하기 문자 줄이다. 그래서, 9자 이상이어야 한다.

INDEXED 자료 집합에서, 열쇠는 자료 집합 안에 있는 기록마디를 지정한다. INDEXED 화일의 열쇠 길이는 선택항 KEYLENGTH에서 지정되어야 한다. 이 지정은 0보다 크고 256보다 작아야 한다. 열쇠는 문자-줄 꼴로 지정되어야 한다.

〔 6 〕 그 밖의 자료 집합 취급 선택항

자료 집합의 자리정하기 (positioning) : ENVIRONMENT의

LEAVE 선택항은 자료 집합이 닫히거나 여러권의 자료 집합에서 하나의 타래 (reel) 가 끝났을 때 자기 테이프를 되감는 것을 막는다. LEVE 선택항은 자료 집합이 앞으로 읽거나 쓰거나 또는 뒤로 읽기 위해 다시 열릴 때도 통상 채용된다.

이송장치 통제 (移送裝置 統制 Carrage Control) : 두개의 서로 배제하는 ENVIRONMENT 선택항 CTLASA과 CTL 360은 REORD 인쇄기 (printer) 와 RECORD 천공 화일을 위한 주머니 선택 (staker selection) 에서 이송장치 통제에 쓰인다.

완충역 할당 (緩衝域 割当 buffer allocation) : 완충역은 자료 집합으로 또는 그로부터 전송되는 자료의 임시 기억소로 사용되는 내부 프로그램 기억역이다. 한개 자료 집합을 위한 두개 완충역의 할당은 입력과 출력 활동을 계속해서 일어나게끔 허용한다.

선택항 BUFFERS(n) 은 자료 집합에 할당되는 완충역의 수 (n) 을 지정한다. D - 편성자에서. n은 1 또는 2이다. BUFFERS(n) 선택항은 UNBUFFERED 화일에는 사용되어서는 안된다. 만일 BUFFERS(n) 선택항이 지정되지 않으면, 완충역 수는 1로 대행된다.

INDEXED DIRECT UPDATE 화일에서, ADDBUFF(n) 선택항은 추가되는 기록마디를 위해 확보되는 추가되는 완충역을 지정하는데 사용될 수 있다. n은 확보되는 완충역 기억소의 글자 수를 지정하는 부호 없는 십진 고른수 상수를 표현한다 ; 이는 32768보다 작아야 한다. INDEXED DIRECT UPDATE 화일에 기록마디를 추가되는 동안, 이는 새로운 기억마디가 열쇠로 지정된 자리에 끼워지기 전에, 해당 궤도에 관계되는 기록마디를 (블러크 되었던 또는 블러크 안되었던) 옮기는데 필요하다. 그러나, 이는 각 기

록마디를 단순히 읽고 쓰는 필요를 없애므로써 시간이 절약될 수 있다. ADDBUFF(n) 선택항은 빠른 속력으로 찾을 수 있게끔 충분한 여백을 확보케 한다.

명찰 없는 테이프 처리 (precessing unlabeled tape) : 이는 명찰이 없거나 비표준 명찰 (非標準 名札 non-standard label) 을 가진 자기 테이프를 읽고 쓰는데 요망된다. NOLABEL 선택항이 명찰 처리가 없음을 가리키는데 사용된다. 출력에서, NOLABEL에 추가해서 NOTAPEMK 선택항이 지정되지 않는한 테이프의 첫번째 기록마디로서 테이프 자국 (tape mark) 이 자동으로 기록된다.

NOTAPEMK는 완충역 없는 (unbuffered) 화일에는 허용되지 않는다. 앞서는 테이프자국 (leading tapemark) 이 없는 테이프에 있는 자료 집합은 UNBUFFERED 속성을 가진 INPUT 화일로서는 읽혀지지 않는다.

VERFY 선택항 : 이는 기록마디가 쓰여질 때 그 기록마디가 제대로 쓰여졌는가를 검토할 때 요망된다. ENVIRONMENT 속성에 있는 VERIFY 선택항은 기록 연산 뒤에 읽기 검토를 하게 한다. 이 선택항은 직접 접근 기억 장치와 연관된 화일에만 허용된다.

EXTENTNUMBER 선택항 : EXTENTNUMBER 선택항은 REGIONAL 또는 INDEXED 화일에 사용되는 범위 (範囲 extents) 의 수를 지정하는데 사용된다. REGIONAL 화일에, EXTENTUNMBER는 선택적이고, INDEXD 화일에서 이는 지정되어야 하고 모든 자료 지역 범위를 포함해야 한다. 이는 주 원통 색인 범위 (master cylinder index extent) 와 모든 종속되는 넘치기 범위 (overflow extent) 도 포함함을 말한다.

INDEXED 화일과 연관된 그 밖의 ENVIRONMENT 선택항 : 주 색

인 (master index 主 索引 master index) 이 있음을 지정하기 위해서는, INDEXMULTIPLE 선택항이 사용된다.

HIGHINDEX 선택항은 장치 형식이 **MEDIUM** 선택항에서 지정된 바와 다를 경우에 높은 수준 색인 또는 색인들 (high index or indexes) 이 등재되어 있는 장치의 형식을 지정할 때 사용된다.

OFLTRACKS 선택항은 추가하는 기록마디를 위해 각 원통 (cylinder) 에 확보되는 궤도의 수를 지정하는데 사용된다. OFLTRACKS 지정은 INPUT 화일에는 의미가 없다.

INDEXED DIRECT 화일을 위한 INDEXAREA는 원통 색인 또는 원통 색인의 일부가 내부 기억소에 있음을 지정하는데 사용된다. 보기로, 내부 기억소에 있는 원통 색인 가재난 (cylinder index entry) 의 수가 자료 집합에 있는 원통의 수와 같다면, READ/WRITE/REWRITE 문은 내부 기억소에 있는 원통 색인을 찾을 것이며 직접 접근 장치를 찾지 않을 것이다.

ENVIRONMENT 선택항에 대한 더 상세한 지식은 제Ⅱ부, 제9장 "속성" 과 부록 가를 참조하라.

제4절 자료 전송 (資料 転送 data transmission)

이 장 앞에서 논술된 바와 같이, PL/I은 두가 자료 전송 형식 (形式 type) 을 갖추고 있다. 흐름-지향 (stream-oriented) 과 기록마디-지향 (record-oriented)

흐름-지향 전송 (흐름-指向 転送 stream-oriented transmission) 에서, 자료 집합은 문자 꼴로 된 자료 항목의 연속적 흐름이라 여겨진다; 내부 비트-줄 표현과 규약 산수 자료의 내부 형식은 줄에서 나타나지 않는다. 자료 항목은 줄에서 프로그램

변수로 배치되거나 또는 프로그램 변수로 부터 (또는 식으로부터) 줄로 배치된다. 이때 필요한 변환이 일어난다. 흐름-지향 전송문은 기록마디 간의 경계를 무시한다.

기록마디-지향 전송에서 (record-oriented transmission) 에서 자료 집합은 논리 기록마디의 집합체로 취급되며, 각각은 하나 이상의 항목으로 이루어진다. 자료 항목은 계산기에 허용되는 내부. 외부의 어떤 표현도 가질 수 있다. 그리고 자료 변환은 없다. 각 논리 기록마디는 프로그램 변수 또는 완충역으로 또는 거기로부터 하나의 단위로서 전송된다.

동일 자료 집합에 대해서 때에 따라 흐름 전송 (stream transmission) 이나 기록마디 전송 (record transmission) 으로 처리되는 것이 가능하다 ; 그렇지만, 자료 집합은 흐름 전송에서 허용되는 문자 꼴로 되어 있어야 한다. 속성 STREAM이나 RECORD 중의 하나가 어떤 전송 방법이 적용될 것인가를 결정한다

제 5 절 흐름-지향 전송

PL/I 서브세트 언어에서, 두가지 형태의 흐름 전송이 있다 : 나열-지시와 편집-지시 (羅列-指示와 編輯-指示 list-directed and edit-directed) 두가지 형태는 모두 입력과 출력에서 동일문을 사용한다. GET와 PUT. 이들 문은 보통 다음의 정보를 요구한다.

① 자료가 취득되거나 자료가 배치되는 자료 집합과 연관된 화일의 이름.

② 입력 동안 자료 항목이 배치되거나 출력동안 자료 항목이

취득되는 프로그램 변수의 나열. 이 나열을 자료 나열(data list)이라 부른다. 출력에서, 자료 나열은 또 상수와 그 밖의 식을 포함할 수 있다.

③ 편집-지시 전송에서. 흐름에 있는 각 자료 항목의 서식 (書式 format).

만일 화일 이름이 지정되지 않으면, 표준 화일의 하나가 대행된다.

〔 1 〕 나열-지시 전송(羅列-指示 転送 list-directed transmission)

나열-지시 전송은 서식 없이 자료가 전송되는 변수를 지정함을 허용한다.

입력 : 흐름에 있는 자료 항목은 음양 부호가 붙을 수 있는 상수 꼴이다. 자료가 배치되는 변수는 자료 나열(data list)에 의해 지정된다. 항목은 쉼표와/또는 빈자로 구분된다.

출력 : 전송되는 자료는 자료 나열에 의해 지정된다. 흐름에 놓아질 자료의 꼴은 자료 값과 속성을 가진다. 항목은 빈자(blank)로 구분된다. 앞서는 0은 지워진다. 그렇지만, 1보다 작은 값은 소수점 왼쪽의 한개 0과 소수점 이하가 전부 인쇄된다. 이진 고정점수와 부동점수 항목은 전송되기 전에 십진수로 변환된다.

PRINT 화일에서, 자료 항목은 자동으로 실무상 정해진 자리에 할당된다. D-수준 편성자에서, 이들 자리는 1, 25, 49, 73, 97, 121이다. 이들 구획은 프로그래머에 의해 바뀔 수 있다.
(이에 대해서는, IBM 조직/360 Disk and Tape 운영조직, PL/I Pregramer's Guide, order NO. GC24-9005를 보라)

〔2〕 편집-지시 전송(編輯-指示 転送 edit-directed transmission)

편집-지시 전송은 자료가 배치되는 변수 또는 전송될 자료를 지정케 한다. 편집-지시 전송은 외부 매체에 있는 각 항목을 위한 서식(format)을 지정케 한다.

입력: 흐름에 있는 자료는 문자의 연속한 줄이다; 이웃한 항목이 구분되지 않는다. 자료가 배치되는 변수는 자료 나열에 의해 지정된다. GET 문에서 서식 나열(format list)에 있는 서식 항목은 각 변수에 배치될 문자 수를 지정하고 자료의 특성(보기로, 소수점의 위치)을 기술한다.

출력: 전송되는 자료는 자료 나열에 의해 지정된다. 자료가 흐름에서 차지할 서식은 서식 나열에 의해 정해진다.

1. 흐름 전송을 위한 자료 지정(data specification for stream transmission)

자료 지정이(資料 指定 data speification) GET와 PUT 문에서 전송될 자료를 나타내기 위하여 주어진다.

〔1〕 자료 나열(資料 羅列 data list)

나열-지시와 편집-지시 자료 지정은 전송될 자료 항목을 지정하기 위한 자료 나열을 요구한다.

일반형식: (자료-나열)

여기서 자료-나열은 다음과 같다:

요소, 〔, 요소〕...

구문 규칙(構文 規則 syntax rules):

요소의 성질은 자료 나열이 입력 또는 출력에 사용되었느냐에 좌

우된다. 규칙은 다음과 같다.

① 입력에서, 나열-지시와 편집-지시 전송을 위한 자료-나열 요소는 다음의 하나이다 : 요소, 배열, 구조체 변수, 구조체나 배열을 나타내지 않는 유사 변수, 또는 이들 요소를 포함한 반복 지정 (DO 모임의 반복 지정과 비슷함).

② 출력에서, 자료-나열 요소는 다음의 하나이다 : 요소 식, 배열 변수, 구조체 변수 또는 이들 요소를 포함한 반복 지정.

③ 자료 나열의 요소는 산수, 수치 문자, 또는 줄 자료 형식.

④ 일반 형식에서 보인 것처럼, 자료 나열은 괄호에 싸여야 한다.

반복 지정 (反複 指定 repetitive specification)

반복 지정의 일반 서식은 도해 8-1에 있다.

구문 규칙 :

① 반복 지정의 요소 나열에 있는 요소는 앞서 열거한 자료-나열 요소에서 허용된 어느것도 된다.

② 지정에 있는 식은, DO 문에 있는 그것과 같음, 다음과 같다.

ⓐ 지정에 있는 각 식은 요소 식이어야 한다.

ⓑ 지정에서, 식-1은 통제 변수 (統制 變數 control variable)의 시작하는 값을 나타낸다. 식-3은 반복 지정에 있는 자료-나열 요소가 반복된 뒤에 통제 변수에 더해지는 증분 (增分)을 나타낸다. 식-2는 통제 변수가 끝나는 값을 나타낸다. 식-4는 반복 변수를 통제하는 두번째 조건을 나타낸다. 지정의 정확한 뜻은 동일한 지정을 가진 DO 문의 그것과 같다.

마지막 지정이 끝나면, 통제는 자료 나열에 있는 다음 요소로 간다.

③ 각 반복 지정은 괄호에 싸여야 한다. 만일 자료-나열이 반복 지정 하나뿐이라 자료-나열은 두 쌍의 괄호에 싸인다.

④ 도해 8-1에서, 반복 의 "지정" 부분은 여러번 반복될 수 있다.

보기 : DO I = 1 TO 4, 6 TO 10 ;

반복 지정은 풀어질 수 있다; 이는, 반복 지정에 있는 요소가 그 자체로 반복 지정이 될 수 있다. 각각의 DO 부분은 괄호에 싸여야 한다.

보기 :

```
GET EDIT(((A(I,J) DO I = 1 TO 2) DO J = 3 TO 4 ))
        (서식-나열) ;
```

위에 것은 다음과 등가이다.

```
DO  J = 3  TO  4 ;
DO  I = 1  TO  2 ;
GET EDIT ( A ( I, J )) (서식-나열) ;
END ;
END ;
```

A를 첨자로 표시하면 다음 순서이다.

A (1,3), A (2,3), A (1,4), A (2,4)

주 : DO 열의말이 반복 지정에서 사용되지만, 대응하는 END 문은 허용 안된다.

〔 2 〕 자료-나열 요소의 전송

만약 자료-나열 요소가 배열 변수이면, 배열의 요소는 행대순(行大順 row-major order)으로 전송된다. 이는 배열의 가장 오른쪽 첨자가 더 자주 변함을 말한다.

(요소 〔 , 요소 〕 . . . DO 변수 =지정〔 , 지정 〕 . . .)

"지정"은 다음 형식을 갖는다 :

식-1 { TO 식-2 〔 BY 식-3 〕 / BY 식-3 〔 TO 식-2 〕 } 〔 WHILE (식-4) 〕

도해 8-1 반복 지정의 일반 형식

만일 자료-나열 요소가 구조체 변수이면, 구조체의 요소는 구조체 선언에서 지정된 순서로 전송된다.

보기 :

```
DCL 1 A, 2 B(10), 2 C(10) ;
PUT FILE(X) EDIT(A) (서식-나열) ;
```

이는 다음 순서이다 :

B(1), B(2), . . . B(10), C(1), C(2), . . . C(10)

입력에서 새로운 값이 변수에 들어간다.

```
GET EDIT ( N, (X(I) DO I=1 TO N ), J,SUBSTR(NAME,
J,3)) (서식-나열) ;
```

이 문이 실행되면, 자료는 전송되어 각 변수에 배치된다.

2. 나열-지시 자료 지정 (list-edited data specification)

입력과 출력의 나열-지시 자료 지정을 위한 일반 형식은 다음과 같다.

LIST (자료-나열)

열쇠말 LIST는 나열-지시 방식의 전송을 지정하는데 나타나야 한다. 자료 나열은 이 장 앞에 있는 "흐름 전송을 위한 자료 지정"에 기술되었다.

〔 1 〕 흐름의 나열-지시 자료 (list-directed data in the strem)

입력이나 출력이건 흐름의 자료는 다음과 같은 일반적 꼴의 하나이다.

〔 + 1 - 〕 십진-산수-상수 (보기, - 36.7, 108, 32.6 E 10)

문자-줄-상수 (보기, 'ABCD')

비트-줄-상수 (보기, '1011'B)

영국돈 상수는 사용 안된다. 중복과 줄 반복 계수는 허용 안된다. 덧붙인 빈자는 허용된다.

〔 2 〕 나열-지시 입력 형식

자료 이름이 배열이면, 첫번째 상수는 첫번째 배열 요소에 배치되고, 다음 상수는 다음 요소로 배치된다. 배열은 행대순이다.

자료 나열에 있는 구조체 이름은 구조체 서술에 있는 지정된 순서로 포함된 요소 변수와 배열을 열거한 것을 나타낸다.

입력에서, 흐름에 있는 자료 항목은 각각 하나 이상의 빈자가 뒤에 붙거나 몇개의 빈자로 싸인 쉼표가 뒤에 붙거나 해야 한다

빈 난(null field)이란 것은 첫자가 쉼표거나 몇개의 빈자가 들어 있는 한 쌍의 쉼표다. 이 빈 난은 연관된 자료 항목이 바뀌지 않고 남아 있음을 가리킨다.

입력에서 상수의 전송은 자료 나열이 다 되거나 화일끝(end-of-file) 조건이 올때 끝난다. 앞의 경우에서 화일은 전송되거나 건너뛴 마지막 자료 항목을 구분하는 빈자나 쉼표 다음 문자에 자리를 정한다.

문자-줄 상수(character-string constants) : 만일 자료가 문자-줄 상수라면, 둘러 막은 따옴표는(apostrophes) 제쳐놓고 둘러싸인 문자는 문자 줄로 해석된다. 두개의 둘러싸인 따옴표는 하나로 해석된다.

비트-줄 상수(bit-string constants) : 만일 자료가 비트-줄 상수이면, 둘러막은 따옴표와 문자 B는 무시되고 둘러싸인 문자 (0 또는 1)가 비트 줄로 해석된다.

십진-산수 상수(decimal-arithmatic constants) : 만일 자료가 십진-산수 상수이면, 이는 상수에 의해 제공된 기수, 척도, 정도를 가진 규약 산수로 먼저 변환된다. 그런다음 이는 자료 나열에 있는 대응하는 자료 항목에 의해 제공된 내부 표현으로 변환된다.

〔 3 〕 나열-지시 출력 형식(list-directed output format)

자료 나열에 있는 요소 변수의 값은 문자 표현으로 변환되고 흐름으로 전송된다. 빈자가 전송되는 자료 항목의 끝으로 사용된다. 길이는 자료 항목의 내부 정도와 값에 의한다. 규약 산수 자료의 외부 꼴은 부호가 있을 수 있는 십진 상수이다.

규약 고정점 십진수 자료. 자료 항목은 정도(p, q)에 척도

난이 (scaling field) 있으면 그것을 더한 것으로 변환된다. 이는 넓이 W의 난에 만약 있다면 척도 난을 더한 것으로 전송된다.

만일 q가 0보다 크거나 같고 p보다 작거나 같으면, W=P+3이고 d=q이다. 여기서 d는 소수 자리 수를 나타낸다. 만일 q가 0보다 작거나 p보다 크면, W=P+3+n이다. 여기서 n은 p를 표시하는데 드는 자리 수이다. 출력에서 값 0의 표현은 아래 식으로 정해진다. (여기서 p는 십진 정도이고, q는 십진 척도 인자(scale factor)이고, W는 난의 넓이다)

고른수 (q=0)는 P+2개 빈자가 앞선 하나의 0으로 표시된다.

$\underbrace{bbb\ldots b\emptyset}_{P+2}$ W=P+3

소수를 가진 수 (0 < q ≤ P)는 소수점 뒤에 q개 0과 앞에 P−q+2 빈자로 표현된다.

$\underbrace{bb\ldots bb}_{P-q+2}.\underbrace{\emptyset\emptyset\ldots\emptyset}_{q}$ W = P+3

q < 0 이거나 q > p인 수

$\underbrace{bb\ldots b\emptyset}_{P}F\pm\underbrace{XX}_{q}$ W = P+3+K

q를 표현하는데 드는 K자리

영 지우는 것이 난의 왼쪽에서 이루어진다. 그리고 값이 0보다 작으면, 음 부호가 첫번째 유의 숫자 바로 앞에 온다 (또는 만일 값이 소수이면, 소수점의 왼쪽)

규약 부동점 십진수 자료. 자료 항목은 부동점수 서식 항목을 위한 규칙에 따라 변환된다 : E (W, d, S). E-변환에서,

W = P + 6, d = P − 1, S = P, 여기서 P는 정도 속성이다.

출력에서 값 0의 표현은 다음 식으로 정해진다.

bØ.$\underbrace{ØØ\ldots Ø}_{P-1}$E±ØØ W = P + 6

규약 고정점 이진수 자료. 자료 항목은 흐름에 놓아지기 전에 고정점 십진수로 변환된다.

규약 부동점 이진수 자료. 자료 항목은 흐름에 놓아지기 전에 부동점 십진수로 변환된다.

수치 문자 자료. 수치 문자 자료의 기수는 십진이다. 십진수 자료 항목의 외부 형식과 난 넓이는 모양 지정에서 기술된 것과 같다.

비트-줄 자료. 비트 줄은 0과 1로 되고 따옴표로 싸이고 뒤에 문자 B가 붙는 비트 줄 상수의 문자 표현으로 변환된다.

문자-줄 자료(문자 모양도 포함). 문자 줄은 그대로 써진다. 만일 화일이 속성 PRINT를 가지면, 싸는 따옴표는 공급되지 않는다. 그리고 한개의 따옴표는 변경되지 않는다. 난 넓이는 줄의 현재 길이다. 만일 화일이 속성 PRINT를 갖지 않으면, 싸는 따옴표가 제공되고, 들어 있는 한개의 따옴표는 두개 따옴표로 대치된다. 난 넓이는 줄의 길이 더하기 추가된 따옴표이다.

보기 :

```
LIST ( CARD.RATE,DYNAMICFLOW )
LIST ( ( THICKNESS ( DIS ) DO DIS = 1 TO 1000 ) )
LIST ( P,Z,M,R )
LIST ( A*B/C, ( X+Y ) **Z )
```

마지막 보기의 지정은 출력에서만 사용된다. 그것은 연산 식을

포함하고 있기 때문이다.

다음 보기는 빈 난의 효과를 보여준다. 빈 난은 연관된 나열 항목의 값이 불변인 채로 남아있음을 지정한다. (문자 b는 빈자를 가리킨다)

문 : GET LIST (A,B,C,D) ;

입력 자료 흐름 : b,b3.14b,7.2bb

A , B , C , D는 요소 변수이다. 이 경우 3.14는 B에, 7.2는 D에 대입된다. 변수 A와 C는 불변인 채 남아 있다.

3. 편집-지시 자료 지정 (編輯-指示 資料 指定 edit-directed data specifiation)

편집-지시 자료 지정의 일반 형식은 다음과 같다.

EDIT (자료-나열) (서식-나열) [(자료-나열) (서식-나열)]

. . .

① 괄호에 싸인 자료 나열은 입력에서 하나 이상의 변수, 출력에서 하나 이상의 식으로 되어 있다. 괄호에 싸인 서식 나열은 하나 이상의 서식 항목으로 된다. 세 종류의 서식 항목 (format item) 이 있다 : 줄에 있는 자료를 기술하는 자료 서식 항목 (data format item) , 지면, 줄 (line) , 칸 넌너기 연산을 기술하는 통제 서식 항목 (control format item) , 서식 나열을 가진 독립된 문의 명찰을 지정하는 외딴 서식 항목 (remote format item) . 서식 나열과 서식 항목은 다음에 오는 "서식 나열"에서 상세히 논술될 것이다.

② 입력에서, 흐름에 있는 자료는 각개의 자료 항목으로 독립

안된 문자의 연속적 줄로 여겨진다. 각 자료 항목을 위한 문자 수는 서식 나열에 있는 서식 항목에 의해 지정된다. 문자는 연관된 서식 항목에 따라 처리된다.

③ 출력에서, 자료 나열에 있는 각 항목의 값은 연관된 서식 항목에 의해 지정된 서식으로 변환되고 역시 서식 항목에 의해 길이가 지정된 난으로 흐름에 놓아진다.

④ 입력이나 출력이나, 첫번째 자료 서식항목은 자료 나열에 있는 첫번째 항목에 연관되고, 두번째는 두번째 등으로 연관된다. 만일 서식 나열이 연관된 자료 나열에 있는 항목보다 적은 서식 항목으로 되어 있으면, 서식 나열은 재차 사용된다; 만일 여분의 서식 항목이 있으면, 이들은 무시된다. 서식 나열이 다섯개 자료 서식 항목으로 되어 있고 이와 연관된 자료 나열이 열개 항목으로 되어 있다고 하자. 그러면 자료 나열에 있는 여섯번째 항목은 첫번째 자료 서식 항목과 연관될 것이다. 서식 나열이 열개의 자료 서식 항목으로 되어있고 이와 연관된 자열 나열이 다섯개 항목 뿐이라고 하자. 그러면 여섯번째 부터 열번째 까지는 무시 될 것이다.

⑤ 나열에 있는 배열이나 구조체 변수는 자료 나열에 있는 n개 자료 항목과 등가이다. 여기서 n은 배열이나 구조체에 있는 요소 항목 수이다.

⑥ 만일 자료 나열 항목이 통제 서식 항목과 연관되면, 이 통제 행위가 실행된다. 그리고 자료 나열 항목은 다음의 서식 항목과 짝이 된다.

⑦ 지정된 전송은 마지막 항목이 이와 대응하는 서식 항목을

내

사용해서 처리되면 끝난다. 다음에 오는 서식 항목은, 통제 서식 항목을 포함해서, 무시된다.

⑧ 출력에서, 각개 자료 항목은 이에 대응하는 서식 나열에 의해 지정된 길이를 차지한다.

보기 :

```
GET EDIT(NAME,DATA,SALARY)(A(20),X(2),A(6),F(6,2)) ;
PUT EDIT('INVENTORY='||INUM,INVCODE)(A,F(5)) ;
```

서식 나열 (書式 羅列 format lists)

편집 지시 자료 지정은 서식 나열을 요구한다.

일반 형식 :

(서식-나열)

여기서 서식 나열은 다음과 같다 :

{ 항 목 / n 항목 / n (서식-나열) } [{ 항 목 / n 항목 / n (서식-나열) }] . . .

구문 규칙 :

① 각 "항목"은 아래에 기술한 바 서식 항목을 나타낸다.

② 문자 n은 반복 계수 (iteration factor)를 말한다. 이는 부호 없는 십진 고른수 상수이어야 한다.

일반 규칙 :

세 종류의 서식 항목이 있다 : 자료 서식 항목, 통제 서식 항목 외딴 서식 항목. 자료 서식 항목은 자료 낟이 취하려는 외부 꼴 (external form)을 지정한다. 통제 서식 항목은 지면, 줄, 자리, 건너가기, 자리띄기 (page, line, column, skip, spacing, operation) 연산을 지정한다. 외딴 서식 항목은 (remote item)

블러크 안 어느 곳에서 독립된 FORMAT 문에서 서식 항목을 지정할 수 있게 한다.

여러가지 서식 항목에 관한 상세한 논술은 제Ⅱ부, 제5장, "편집-지시 서식 항목"에 있다.

4. 자료 서식 항목(data format items)

입력에서, 각 자료 서식 항목은 자료 항목과 연관된 문자 수를 지정하고 어떻게 외부 자료를 해석하는 가를 지정한다. 자료 항목은 자료 나열에 있는 연관된 변수에 대입된다. 변수의 속성에 맞게끔 변환되기도 한다. 출력에서, 자료 나열에 있는 연관된 요소의 값은 서식 항목으로 지정된 문자 표현으로 바뀌고 자료 흐름에 들어간다.

네 종류의 자료 서식 항목이 있다 : 고정점수(F), 부동점수(E), 문자-줄(A), 비트(B). 이들은 다음과 같이 지정된다.

F (W[, d[, P]]), E(W , d[, S]), A[(W)], B[(W)]

여기서 문자 W는 난에 있는 문자 수를 지정하는 십진 고른수 상수를 나타낸다. 문자 d는 소수점 오른쪽자리의 수를 지정하다.

세번째 지정 P는 F서식 항목에서 사용된다 ; 이는 척도 인자 (scaling factor)이다. 세번째 지정 (S)는 E 서식에서 부동점수의 첫번째 소난(小欄 subfield)에서 유지되어야 할 자리 수를 지정하는 데에 사용된다. 이들 지정은 제Ⅱ부, 제5장, "편집-지시 서식 항목"에서 상세히 논술된다.

주 : 고정점 이진수와 부동점 이진수 자료 항목은 외부 꼴이 없다. 그러므로 이들은 십진수로 표현되어야 한다. 입력, 출력에서 내부꼴로 변환은 자동으로 된다.

보기 :

① GET FILE(INFILE) EDIT(ITEM) (A(20)) ;

이 문은 INFILE이라는 화일에서 다음 20개 문자를 ITEM에 대입시킨다. 이는 문자-줄 변수이어야 한다. 만일 이것이 문자-줄 변수가 아니라면, 오류가 될 것이다.

주 : 이것이 출력이라면 길이는 없어도 된다. 이때 ITEM 길이가 곧 외부꼴 길이다.

② PUT FILE(MASKFL) EDIT(MASK) (B) ;

MASK는 비트이어야 하고 외부 꼴은 B에 길이 지정이 없으므로 MASK 길이가 된다.

③ PUT EDIT(TOTAL) (F(6,2)) ;

TOTAL이 속성 FIXED(4,2)를 가졌다 하자. 그러면 위 문은 이것이 문자 표현으로 바뀌고 표준 화일로 나간다. 소수점이 마지막 두개 문자 앞에 찍힌다. 6문자란 이 소수점도 포함된다. 앞서는 0는 빈자로 바뀌고, 필요하면 음 부호가 첫번수 문자 앞에 온다.

④ GET FILE(A) EDIT(ESTIMATE) (E(10,6)) ;

소수점은 가수(mantissa 仮数)의 여섯자리 앞에 있는 것으로 된다. 실제의 소수점은 이 가정을 무시한다.

⑤ GET EDIT(NAME,TOTAL) (A(5),F(4,0)) ;

이 문이 실행되면, 표준 입력 화일이 취해진다. 첫번 5 문자는 NAME에 대입된다. 다음 4 문자는 앞서는 그리고/또는 뒤따르는 빈자가 있을 수 있는 산수 문자여야 한다.

5. 통제 서식 항목 (統制 書式 項目 control format item)

통제 서식 항목은 두 종류가 있다 : 자리띄기 서식 항목 (X) (spacing format item (X)) 과 서식 항목 COLUMN,LINE,PAGE, SKIP. X,COLUMN,SKIP은 PRINT와 PRINT 아닌 화일에서 입력, 출력 어디서나 사용된다. LINE,PAGE는 PRINT 화일에서 PUT 문에만 사용된다. PAGE 이외는 보통 십진 고른수 상수를 가진다. LINE, PAGE, SKIP은 또 PUT 문에서 선택항으로 독립되어 나타날수 있다. (SKIP 서식 항목은 GET 문에서도 사용될수 있지만, SKIP 선택항으로는 안된다) 이들이 선택항으로 PUT 문에 나타나면, 식 (expression) 이 십진 고른수 상수 자리에 사용 될 수 있다.

보기 :

① GET EDIT (NUMBER,REBATE) (A(5),X(5),A(5)) ;

표준 화일 SYSIN이 대행되고, X(5)는 5자리 띄는 것을 말한다.

② PUT FILE (OUT) EDIT (PART,COUNT) (A(4),X(2),F(5)) ;

처음 4자리는 PART에서, 다음 두 자리는 빈자로, 다음 5자리는 COUNT에서 온 숫자로 된다.

③ PUT EDIT ('QUARTERLY STATEMENT')
('PAGE, LINE(Z), A(19)) ;
PUT EDIT (ACC#, BOUGHT, SOLD, PAYMENT, BALANCE)
(SKIP(3), A(6), COLUMN(14),F(7,2), COLUMN(30),
F(7,2), COLUMN(45), F(7,2), COLUMN(60), F(7,2)) ;
SKIP(3)은 다음으로 3줄을 가라는 것이다. 고로 2줄의

빈줄이 생긴다. COLUMN(14)는 이 줄의 제 14번째자리 부터 시작하라는 것이다.

PAGE는 다음 지면으로 가라는 것이고, LINE(2)는 이 지면의 2번째 줄로 가라는 것이다.

④ GET EDIT(PART,SEQUENCE) (SKIP,COLUMN(2), A(71), A(80));

GET EDIT(DETAIL)(COLUMN(20),F(15,2));

SKIP 서식 항목은 입력 자료가 다음 줄에서(또는 기록 마디) 자리 잡음을 지정한다.

주 : 통제 서식 항목은 이것이 감지될때 실행한다. 자료 나열이 다한 뒤에 오는 통제 서식 나열은 효과가 없다.

6. 외딴 서식 항목 (remote format item)

외딴 서식 항목 (R)은 다른 곳에 위치한 FORMAT 문의 명찰(또는 명찰 변수)을 지정한다 ; FORMAT 문과 이 외딴 서식 항목을 지정한 GET와 PUT 문은 동일 블러크에 내적이어야 한다

보기 :

```
DCL SWITCH LABEL ;
GET FILE(IN) EDIT(CODE) (F(1));
IF CODE = 1  THEN SWITCH = L1 ;  ELSE SWITCH = L2 ;
GET FILE(IN) EDIT(W,X,Y,Z) (R(SWITCH));
L1 : FORMAT  (4F(8,3));
L2 : FORMAT  (4E(12,6));
```

7. 흐름-지향 자료 전송문 (stream-oriented data transmission statunents)

다음은 STREAM 자료 전송 문의 요약이다. (문은 제Ⅱ부, 제

10장, "문"에서 개별로 상세히 논술된다)

STREAM INPUT :

GET 〔FILE(화일-이름)〕 자료-지정 ;

STREAM OUTPUT :

PUT 〔FILE(화일-이름)〕 자료-지정 ;

STREAM OUTPUT PRINT :

PUT 〔FILE(화일-이름)〕

⎧ PAGE 〔LINE(식) ⎫
⎨ SKIP 〔(식)〕 ⎬ 〔자료-지정〕 ;
⎩ LINE (식) ⎭

"자료 지정"은 만일 통제 선택항의 하나가 나타나면 STREAM OUTPUT PRINT 화일에서만 생략될 수 있다.

위에서, 자료 지정은 다음 꼴을 가진다.

LIST (자료-나열)

EDIT (자료-나열) (서식-나열)
〔(자료-나열) (서식-나열)〕...

서식 나열은 다음의 서식 항목의 어떤 것을 사용해도 된다.

A, B, E, F, R, X	이는 어떤 STREAM에도 사용된다.
PAGE	이는 STREAM OUTPUT PRINT 화일에만
LINE(n)	사용된다.
SKIP〔(W)〕	이는 어떤 STREAM에도 사용된다.
COLUMN(W)	

주 : STREAM 화일에 있는 자료는 사실상 블러크 단위로 전송된다 그래서, STREAM 화일이 닫히면, 마지막 블러크는 이것이 자료로 꽉 채워졌는가 아닌가에 관계없이 전송된다. 이는 자료의 끝이 화일의 끝과 일치하지 않을런지도 모르므로 의외의 결과를 가져올지 모른다. 그러므로 프로그래머는 자료의 끝이 명백히 정해지도록 해야 한다.

제 6 절 기록마디 - 지향 전송 (記録마디 - 指向 転送 record-oriented transmission)

따로 떨어진 기록마디로 되어 있거나 따로 떨어진 기록마디의 집합체로 생성되는 자료 집합은 기록마디 - 지향 연산 문으로 처리될 수 있다. 이들 문은 READ, WRITE, REWRITE, LOCATE이다. 이들 문에 관한 전반적 설명은 이 장에서 구성되어 있다. 이들은 제Ⅱ부, 제 10 장, 〃문〃에서 자세하게 설명된다. 자료 집합에서 얻거나 자료 집합으로 발송되는 각 기록마디는 변수의 자료 속성으로 정의된다. (보통 구조체). 입력 연산에서, 기록마디는 자료 집합으로 부터 얻어내지어 변환 없이 변수에 대입된다. 출력 연산에서, 자료 집합은 자료집합으로 변환 없이 전송된다.

기록마디 전송에 참가한 변수는 수준 1 인 (요소 변수와 배열 변수는 태만에 의해 수준 1 이 된다) 첨자 없는 것이어야 한다. 그리고 어떤 기억소 종류 (storage dass) 도 될 수 있다. 이들은 명찰이나 지침 변수가 될 수 있다. 그러나 이런 자료는 전송에서 자체의 값을 상실할 수도 있다.

RECORD 전송에서, 이는 만일 화일이 BUFFERED 속성을 가지면 완충역 (buffer) 에 있는 기록마디에서 연산을 하는 것이 가능하다 완충역 안에서 하는 연산은 기초된 변수 (hared variable), 이는 기록마디의 자료 속성을 기술한다, 와 지침 변수 (pointer variable) 이는 완충역 안에서 기초된 변수의 위치를 다른 값과 동일하게 놓을 수 있다. 의 사용으로 이루어질 수 있다. 기초된 변수와 이와 연관된 지침 변수는 다음 꼴로 BASED 기억소 종류로 선언된다.

BASED (지침 - 변수)

지침 변수 자체는 BASED 기억소 종류 속성을 갖지 못한다; 태만은 AUTOMATIC 이다. 지침 변수는 INTERNAL이나 EXTERNAL 범위 속성을 가질 수 있고, 태만은 INTERNAL이다. 그러나 기초된 변수의 범위는 항상 INTERNAL이다. 지침 변수는 POINTER 속성을 가지고서 명시 선언되어야 한다.

보 기 :

```
DCL RECID POINTER,
    1  MASTERREIORD BASED(RECID),
    2  IDENTIFICATION CHAR(10),
    2  NAME  CHAR(30), 2 ADDRESS,
    3  STREET CHAR(15), 3 CITY CHAR(15),
    3  STATE CHAR(15), 3 ZIP CHAR(5);
```

이름 MASTERRECORD는 완충역에 위치한 기록마디를 기술하는 데에 사용될 수 있는 기초된 변수이다. 기록마디의 난은 MASTERRECORD로 선언된 속성에 맞아야 한다. RECID는 완충역 안에서 MASTERRECORD의 자리를 지적해 주는 지침 변수이다. 이 지침 변수는 명시 선언되었다.

만일 AUTOMATIC이 아닌 어떤 속성이 지침 변수에 선언되려면 이들은 명시 선언되어야 한다.

보 기 :

```
DCL RECID POINTER STATIC EXTERNAL;
```

기초된 변수를 지정한 입력/출력 연산에서, 지침 변수는 READ 나 LOCATE 문에서 SET 선택항에서 지정되어야 한다.

1. 기록마디-지향 자료 전송 문 (record-oriented data transmission statements)

외부 기억소로 또는 그로부터 실제로 기록마디를 전송케 하는 세개 문이 있다. 이들은 READ, WRITE, REWITE. 네번째 문은 LOCATE, 기억소를 다음의 전송을 위하여 완충역에 할당시킨다. 화일의 속성은 어떤 문이 사용될 수 있는가를 결정한다.

READ 문은 어떤 INPUT나 UPDATE 화일에도 사용될 수 있다. 이는 기록마디를 변수로 직접 또는 완충역으로 자료 집합에서 전송시킨다. 블러크된 기록마디에서, READ 문은 기록마디를 완충역으로 부터 변수로 전송시킨다; 또 만일 SET 선택항이 사용된다면, 이는 지침의 값을 완충역에 있는 논리 기록마디를 가르키게 한다. 블러크된 기록마디에서, 매 READ가 물리적 입력이 되는 것은 아니다.

WRITE 문은 어떤 OUTPUT 화일이나, DIRECT UPDATE 화일에 사용될 수 있으나, SEQUENTIAL UPDATE는 안된다. 이는 기록마디를 자료 집합으로 전송케 한다. 블러크 안된 기록마디에서, 전송은 변수로 부터 또는 완충역으로 부터 직접 전송된다. 블러크된 기록마디에서, WRITE 문은 논리 기록마디를 완충역에 갖다 놓게 한다. 단지 기록마디의 블러크 만들기가 끝났을 때만 실제의 물리 출력이 있다.

REWITE 문은 기록마디를 UPDATE 화일에서 바꾸어 놓는다. SEQUENTIAL UPDATE 화일에서, REWRITE 문은 화일로 부터 읽힌 마지막 기록마디가 고쳐써질 것을 지정한다; 그러므로 기록마디는 이것이 고쳐써지기 전에 읽혀야 한다.

DIRECT UPDATE 화일에서, REWRITE 문은 열쇠(key)를 지정해야

한다; 그러므로 어떤 기록마디도 이것이 읽혔건 아니건 간에 고쳐써질 수 있다.

LOCATE 문은 기초된 변수가 지정된 화일을 위한 출력 완충역에 할당되고 지침(pointer)이 위치를 지적해 주기 위해 세워짐을 지정한다. 기초된 변수와 지침 변수는 다 함께 LOCATE 문에서 지정되어야 한다. 기초된 변수는, 가변 길이 기록마디의 경우, 기록마디의 길이를 결정하기에 사용된다. LOCATE 문은 즉시 자료 전송을 지정하는 것이 아니다; 완충역의 내용은 부정(不定)이다. 값은 기초된 변수에 대입되어야 한다. 기록마디는 다음에 오는 WRITE, LOCATE, CLOSE 문이 실행될 때까지 기록되지 않는다. 블러크된 기록마디의 경우, 뒤따르는 LOCATE 문은 지침을 앞선 기록마디의 완충역에서 위치를 지적해 주도록 세우게할 뿐이다.

〔1〕 기록마디-지향 전송 문의 선택항

기록 마디-지향 전송 문에 허용된 선택항은 연관된 문의 속성과 문의 목적에 따라서 다르다. 각 화일 형식에 허용된 결합의 열거가 이 장 뒤에 나온다. 각 선택항은 값이 뒤따르는 열쇠말로 구성된다. 이 값은 화일 이름, 변수 또는 식이다. 이 값은 언제나 괄호에 싸여야 한다. 어떤 문에서나, FINE 선택항이 먼저 나타나야 한다.

FILE 선택항

화일 선택항(또 FILE 지정이라고도 함)은 모든 기록 마디-지향 문에 나타나야 한다. 이는 화일 이름을 지정한다.

이는 괄호에 싸인 화일 이름이 뒤따르는 열쇠말 FILE로 된다.

INTO 선택항

INTO 선택항은 어떤 형식의 INPUT나 UPDATE 화일에서 READ

문에 사용될 수 있다. INTO 선택항은 논리 기록마디가 대입될 변수를 지정한다. 형식은 기록 마디가 임시인 완충역을 지나면 아니면 간에 같다. 변수는 기초된 변수가 될 수 있다.

READ FILE(DETAIL) INTO(RECORD1);

이는 다음에 오는 기록마-디가 변수 RECORD1에 대입됨을 지정한다.

SET 선택항

SET 선택항은 SEQUENTIAL BUFFERED INPUT나 UPDATE 화일을 위한 READ 문에서 사용될 수 있다. 이는 매 LOCATE 문마다 나타나야 한다. SET 선택항은 완충역에서 논리 기록마디를 가르키는 지침 변수를 지정한다.

READ FILE(MASTER) SET(RECIDENT);

LOCATE PAYREC FILE(PAYROLL) SET(P);

첫번째 보기는 MASTER로 부터 오는 기록 마디가 읽히고 지침 변수 RECIDENT가 완충역에서 그 위치를 가르키도록 세워짐을 지정한다. 만일 논리 기록마디가 블러크된 기록마디의 일부이며, 블러크에 있는 첫번째 기록마디가 아니면, 이 문의 실제 결과는 단순히 지침 값을 세울 것이다. RECIDENT의 값은 기초된 변수와 연관되어야 하고, 그래서 기록마디의 난이 접근될 수 있도록 한다.

두번째 보기는 기초된 변수 PAYREC가 완충역에서 할당되며 이것의 위치가 지침 변수 P에 배당됨을 지정한다.

P는 기초된 변수 PAYREC와 함께 선언되어야 한다. LOCATE문은 언제나 기초된 변수를 지정해야 한다. 기초된 변수의 할당에 따라, 값이 이것에 대입된다. 기록마디는 다음에 오는 WRITE, LOCATE, 또는 CLOSE 문이 화일 PAYROLL을 위해 실행될 때 기

록된다. 만일 기록 마디 PAYREC가 블러크된 기록마디의 일부라면, 다음에 오는 LOCATE 문은 동일 블러크 안에 있는 다음 기록마디를 할당할 뿐이다.

FROM 선택항

FROM 선택항은 어떤 OUTPUT 화일과 DIRECT UPDATE 화일에서도 WRITE 문에 사용되어야 한다. 이는 또 어떤 UPDATE 화일에서도 REWRITE 문에 사용될 수 있다. FROM 선택항은 기록마디가 그것으로 부터 기록될 변수를 지정한다.

```
WRITE FILE(MASTER) FROM(MASREC);
REWRITE FIL(MASTER) FROM(MASREC);
```

양쪽 문이 변수 MASREC의 값이 화일 MASTER에 기록됨을 지정한다. WRITE 문의 경우, 이는 SEQUENTIAL OUTPUT 화일에 새로운 기록마디를 지정한다. REWRITE 문은 MASREC가 SEQUENTIAL UPDATE 화일에서 읽은 마지막 기록마디를 바꿔놓음을 지정한다.

KEY 선택항

KEY 선택항은 REGIONAL 또는 INDEXED 구성의 자료 집합과 연관된 화일에만 사용된다. KEY선택항은 INPUT나 UPDATE 속성을 가진 DIRECT 화일에서 READ와 REWRITE 문에 사용되어야 한다. KEY 선택항은 INDEXED 자료 집합 구성에서 INPUT나 UPDATE 속성을 가진 SEQUENTIAL 화일의 READ 문에 사용될 수 있다. KEY 선택항이 사용되는 어떤 화일도 KEYED 속성을 가져야 한다.

만일 REWRITE 문이 블러크된 INDEXED 화일에 사용되면, 사용자는 기록마디의 열쇠 부분이 바뀌지 않도록 해야 한다.

만일 INDEXED 자료 집합이 순차로 읽히면, KEY 선택항은 특정 기록마디에 자료 집합을 갖다 놓는데 사용될 수 있다. 다음에 오는 KEY 선택항 없는 문은 순차 읽기를 그 점부터 계속하게 한다. INDEXED 출력 화일에 있는 열쇠는 오르막 순이어야 한다

KEY 선택항은 괄호에 싸인 요소 식을 뒤에 달은 열쇠말 KEY로 이루어진다. 이 식은 특정 기록마디를 가르키는 원시 열쇠이다. 요소 식은 REGIONAL(1)에서 8 자리이고 REGIONAL(3)과 INDEXED에서 KEYLENGTH에 의해 지정된 길이의 문자 줄로 변환된다.

다음은 문자 줄과 적용되는 자료 집합 구성을 위해 표현되는 것의 요약이다.

REGIONAL(1)

요구되는 기록마디의 상대적 기록마디 번호를 지정하는 8 자리의 문자.

REGIONAL(3)

오른쪽 8 자리가 숫자로 되어있는 문자 줄. 이들 오른쪽 8 문자는 찾아야 할 지역인 상대 궤도를 지정한다. 접근될 기록마디는 KEYLENGTH에 의해 지정된 길이의 문자 줄로 바뀌어지는 원시 열쇠와 정확히 일치하는 기록된 열쇠로 지적된다. 이 줄은 언제나 지역을 지적하는 오른쪽 8 자리를 포함하고 있다.

주 : 기록된 기록마디나 궤도 번호는 2^{24} 보다 작아야 한다.

INDEXED

문자 줄, 첫번째 n개는 기록마디의 기록된 열쇠와 정확히 같아야 한다 (여기서 n은 KEYLENGTH에 의해 지정된 수 이다.

만일 기록된 열쇠가 기록마디에 들어있고 기록마디가 블러크 되었다면, 기록마디 안에 있는 열쇠 위치는 ENVIRONMENT 속성의 KEYLOC(n) 선택항에서 지정되어야 한다.

외 KEY선택항에 있는 식은 유효한 열쇠가 되어야 한다.

3 보 기 :

```
READ FILE(MASTER) INTO(MASREC) KEY('ØØØØ3253');
READ FILE(FILEX)  INTO(ORDERREC) KEY(NAMEIIAREA #);
READ FILE(MASTER) INTO(PAYREC) KEY(NAME);
```

첫째 문은 REGIONAL(1) 자료 집합에서 기록마디 번호 3253이 읽혀지고 변수 MASREC에 대입될 것임을 지정한다.

둘째 문은, 이는 REGIONAL(3) 자료 집합에 알맞을 것이다. 기록마디가 읽혀지고 ORDERREC 변수에 들어감을 지정한다.

기록마디는 AREA #에 의해 지적된 지역에서 발견될 것이다; 해당 기록마디는 원시 열쇠와 꼭 같은 기록된 열쇠에 의해 인지된다. 변수 AREA #는 8 자의 문자-줄을 나타내야 함을 유의하라,

세번째 문은, 이는 INDEXED 자료 집합에 알맞을 것이다. 변수 NAME의 값과 동일한 열쇠를 가진 화일 MASTER의 기록마디가 변수 PAYREC에 읽혀들어감을 지정한다.

KEYFROM 선택항

KEYFROM 선택항은 REGIONAL 또는 INDEXED 자료 집합을 새로 만들려고 사용되는 WRITE 문에서 또는 INDEXED나 REGIONAL 자료 집합에 새 기록마디를 추가하려고 사용되는 WRITE문에서 지정되어야 한다. 이는 CONSECUTIVE 구성에는 사용되지 못한다.

그러므로, 이는 SEQUENTIAL OUTPUT 화일 또는 DIRECT OUTPUT 또는 DIRECT UPDATE 화일에서 WRITE 문에 나타날 수 있다. KEYFROM 선택항을 가진 어떤 화일도 KEYED 속성을 가져야 한다. 만일 WRITE가 INDEXED 구성을 가진 자료 집합에서 실행되고 만일 KEYLOC가 지정되면, KEYFROM 선택항에서 지정된 열쇠 값은 KEYLOC에 의해 지정된 기록마디에 있는 자리로 옮겨간다.

KEYFROM 선택항은 기록마디가 기록될 자료 집합 안의 위치를 지정한다. REGIONAL(1) 자료 접합에서, 이는 단지 지역 번호를 지정한다. REGIONAL(3) 자료 집합에서, 이는 또 기록된 열쇠로서 기록될 문자 줄을 지정한다. INDEXED 자료 집합에서, 이는 기록된 열쇠를 지정하며, 그 값은 위치를 결정하는데 사용된다. 이는 괄호에 싸인 식을 뒤에 가진 열쇠말 KEYFROM으로 써진다. 이 식은 상수, 변수, 또는 어떤식도 된다. 이 값은 항상 문자 줄로 변환된다. REGIONAL(1)을 제하고, KEYLENGTH 선택항이 써질 기록된 열쇠의 길이를 지정해야 한다.

보 기:

```
WRITE FILE(PAYROLL) FROM(PAYREC) KEYFROM(NAME||TRACK);
```

위 문은, 이는 REGIONAL(3) 자료 집합에 알맞다. PAYREC의 값이 다음 순차 기록마디로서 써질 것을 지정한다.
TRACK의 값이 기록마디가 써지는 지역을 지정한다. 원시 열쇠는 NAME의 값과 TRACK의 값이 이어지고, 이것이 기록된 열쇠로 써진다.

KEYTO 선택항

KEYTO 선택항은 INDEXED 자료 집합과 연관되어 있는 SEQUENTIAL INPUT 또는 UPDATE 화일에서 DEAD 문에 사용될

수 있다. 이는 열쇠말 KEYTO를 사용해서 지정된다. 이 KEYTO는 배열이나 구조체가 아닌 문자-줄 변수의 이름이 괄호에 싸여 뒤따른다. 이는 기록된 열쇠가 문자 변수에 대입됨을 지정한다. KEYTO 선택항이 지정된 어떤 화일도 KEYED 속성을 가져야 한다.

보 기 :

```
READ FILE(DETAIL) INTO(INVTRY) KEYTO(KEYCHK);
```

〔2〕 기록마디-지향 전송 문 형식

다음은 화일 속성별로 선택항에 따른 허용된 RECORD 전송 문의 요약이다.

SEQUENTIAL BUFFERED INPUT (연속(consecutine)):

```
READ FILE ( 화일-이름 )  INTO ( 변수 ) ;
READ FILE ( 화일-이름 )  SET ( 지침-변수 ) ;
```

SEQUENTIAL BUFFERED OUTPUT (연속) :

```
WRITE FILE ( 화일-이름 )  FROM ( 변수 ) ;
LOCATE 변수  FILE ( 화일-이름 )  SET ( 지침-변수 ) ;
```

SEQUENTIAL BUFFERED UPDATE (연속) :

```
READ FILE ( 화일-이름 )  INTO ( 변수 ) ;
READ FILE ( 화일-이름 )  SET ( 지침-변수 ) ;
REWRITE FILE ( 화일-이름 ) ;
REWRITE FILE ( 화일-이름 )  FROM ( 변수 ) ;
```

SEQUENTIAL UNBUFFERED INPUT (연속) :

```
READ FILE ( 화일-이름 )  INTO ( 변수 ) ;
```

SEQUENTIAL UNBUFFERED OUTPUT (연속) :

```
WRITE FILE ( 화일-이름 )  FROM ( 변수 ) ;
```

SEQUENTIAL UNBUFFERED UPDATE (연속) :

READ FILE (화일 - 이름) INTO (변수) ;

REWRITE FILE (화일 - 이름) FROM (변수) ;

SEQUENTIAL INPUT (색인 (indexed)) :

READ FILE (화일 - 이름) INTO (변수)
KEYTO (문자 - 변수)
KEY (요소 식)

SEQUENTIAL OUTPUT (색인) :

WRITE FILE (화일 - 이름) FROM (변수)
KEYFROM (요소 식)

SEQUENTIAL UPDATE (색인) :

READ FILE (화일 - 이름) INTO (변수)
KEYTO (문자 - 변수)
KEY (요소 식)

REWRITE FILE (화일 - 이름)
FROM (변수)

DIRECT INPUT (색인, 지역 (regienal)) :

READ FILE (화일 - 이름) INTO (변수) KEY (식) ;

DIRECT OUTPUT (지역) :

WRITE FILE (화일 - 이름) FROM (변수) KEYFROM (요소 식) ;

DIRECT UPDATE (색인, 지역) :

READ FILE (화일 - 이름) INTO (변수) KEY (식) ;

REWRITE FILE (화일 - 이름) FROM (변수)
KEY (요소 식) ;

WRITE FILE (화일 - 이름) FROM (변수)
KEYFROM (요소 식) ;

〔 3 〕 기록마디-지향 전송의 요약

다음은 RECORD 전송의 사용에서 고려점을 망라한 것이다

① SEQUENTIAL 화일은 자료 집합 기록마디의 접근, 만들기, 수식이 특정 순서로 이루어짐을 지정한다. 이 순서란 자료 집합의 첫째 기록마디로 부터 마지막으로 가는 순서이다 (또는 만일 BACKWRDS 속성이 지정되었으면 그 반대이다)

② DIRECT 화일은 자료 집합 기록마디의 접근, 만들기, 수식이 임의 순서 (random order)로 이루어질 수 있음을 지정한다. 연산될 특정 기록마디는 지정된 열쇠에 의해 지정된다

③ SEQUENTIAL 접근 방법으로 접근되고 만들어지고 수식되는 자료 집합은 기록된 열쇠를 가질 수도 안가질 수도 있다. 만일 있다면, 이 열쇠는 순차로 접근될 때는 무시되고, 또는 KEYTO와 KEYFROM 선택항에 의해 자료 집합에서 끄집어내지거나 자료 집합으로 놓아진다. 기록된 열쇠를 가진 자료 집합을 만드는데 가장 효율적 방법은 SEQUENTIAL OUTPUT 화일로 하는 것이다. 이는 다음에 DIRECT 화일로서 접근될 수 있다.

④ SEQUENTIAL INPUT와 SEQUENTIAL UPDATE 화일은 요구되는 기록마디의 열쇠를 지정하는 READ 연산에 의해 자료 집합 안에서 특정 기록마디에 자리잡을 수 있다. 그런뒤에 KEY 선택항 없는 뒤따르는 READ 문은 자료 집합의 그 점에서 계속하는 순차 읽기를 하게 한다. 이런 종류의 접근은 만일 자료 집합이 INDEXED 구성이고 그 화일이 KEYED 속성을 가질 때만 사용될 수 있다.

⑤ SEQUENTIAL UPDATE 화일에서 자료 집합에 있는 기록마디

는 고쳐 써지거나, 수정되거나 무시될 수 있다. 그러나 기록마디의 수는 늘어나거나 줄어들지 못한다. UPDATE 화일에 있는 기록마디는 REWRITE 문을 사용해서 바꿔놓을 수 있다.

⑥ SEQUENTIAL UPDATE 화일에서 REWRITE 문에 있는 FROM 선택항은 자료가 거기로 부터 나와 고쳐 써질 변수의 이름을 가져야 한다.

⑦ 만일 READ INTO 선택항이 SEQUENTIAL BUFFERED UPDATE 화일을 인용하는데 사용되고 다음에 오는 REWRITE 문이 FROM 선택항을 사용하지 않으면, 자료 집합에 있는 기록마디는 완충역에 있는 것으로 바꿔지고 READ 문의 INTO 선택항에서 지정되었던 변수에 있는 것으로 바꿔지지 않는다

⑧ WRITE 문은 자료 집합에 기록마디를 추가하고, 반면 REWRITE 문은 기록마디를 바꿔 놓는다. 그래서 WRITE 문은 OUTPUT 화일과 DIRECT UPDATE 화일에 사용될 수 있고, REWRITE 문은 UPDATE 화일에만 사용될 수 있다 또 DIRECT 화일에서, REWRITE 문은 바꿔놓아질 기록마디를 지적하기 위하여 KEY 선택항을 사용한다 ; WRITE 문은 KEYFROM 선택항을 사용한다. 이 선택항은 기록마디가 쓰여질 장소만이 아니라 REGIONAL(1)을 빼고는 기록될 열쇠를 가리킨다.

제 9 장 편집하기와 줄 취급
(編輯하기와 줄 取扱
editing and string handling)

산수, 비교, 비트 줄에 의해 되는 자료 취급은 줄 취급과 편집 설비까지 확장된다. 이들 설비는 속성, 문 선택항, 내조립 함수, 유사 변수에 의해 지정 된다.

제 1 절 대입에 의한 편집

대입 문에 의해서 되는 편집의 가장 기본적 꼴은 등호의 오른편의 자료 형식을 받아드리는 변수의 속성에 맞추어 변환시키는 것도 포함한다. 자료 변환에 관한 규칙은 제 4 장, 〃식〃과 제 Ⅱ 부, 제 6 장, 〃자료 변환〃에서 논술된다.

1. 줄 자료의 길이 바꾸기

어떤 값이 줄 변수에 대입될 때, 필요하면, 이는 받아드리는 줄 형식으로 변환되고, 또 필요하면, 받아드리는 줄의 길이에 맞게 하려고 오른편이 잘리거나 늘어난다.

보 기 :

```
SUBJECT → CHAR(10)이라 한다.
SUBJECT = 'TRANSFORMATIONS';
SUBJECT = 'TRANSFORMA';
```

첫번 문은 등호의 오른편의 오른편 5문자가 잘려 대입되어 두번째 문과 효과가 같다.

```
SUBJECT = 'PHYSICS':
SUBJECT = 'PHYSICSbbb';
```

b는 빈자를 표시함. 첫번째 문은 두번째 문과 등가이다.

CODE → BIT(10)이라 한다.

CODE = '110011'B;

CODE = '1100110000'B;

첫번째 문은 두번째 문과 등가이다.

SUBJECT = '110011'B;

SUBJECT = '110011bbbb';

SUBJECT = '1100110000'B;

SUBJECT = '1100110000';

첫째와 둘째, 셋째와 넷째는 각각 등가이다.

2. 그 밖의 대입 물

대입 문에 추가해서, PL/I은 변수에 값을 대입하는 또 다른 두가지를 준비하고 있다.

〔1〕 입력과 출력 연산

흐름-지향 연산은 자료 항목이 읽히거나 쓰여질 때 응용되는 여러가지 편집 기능을 보유하고 있다. 이 편집 기능은 대입 문에 의해 준비된것과 비슷하나, 입력에서 문자 형식으로 부터 변환되고 출력에서 문자 형식으로 변환되는 것만이다. 즉 외부 매체에 있는 형식은 문자이다.

〔2〕 GET와 PUT문에서 STRING 선택항

GET와 PUT문의 STRING 선택항은 내부 기억소간에 자료를 전송 시킨다.

FILE선택항은 제거된다.

GET STRING (문자-줄-변수) 자료-지정;

PUT STRING (문자-줄-변수) 자료-지정;

GET 문은 자료 나열에 있는 변수에 대입될 자료 항목이 지정된 문자-줄 변수에서 취해질 것임을 지정한다. PUT 문은 자료 나열의 자료 항목이 지정된 문자를 변수에 대입됨을 지정한다.

〃자료 지정〃은 통제 서식 항목 PAGE, SKIP, LINE, COLUMN 이 사용되지 못한다는 점을 빼고는 입력과 출력에서 기술된 바와 같다. 이는 다시 두가지 형식을 갖는다.

LIST (자료-나열)

EDIT (자료-나열) (서식-나열)

나열-지시 GET 문에서, 문자 줄에 있는 각개 항목은 쉼표나 빈 자로 떨어져 있어야 한다.

보 기 :

```
PUT   STRING(RECORD) EDIT(NAME,PAY#,HOURS*RATE)
      (A(12), A(7), F(8));
READ FILE(INPTR) INTO(TEMP);
GET   STRING(TEMP)  EDIT(CODE) (F(1));
IF    CODE ¬=1 THEN GO TO OTHERTYPE;
GET   STRING(TEMP) EDIT(X,Y,Z)(X(1),3F(10.4));
PUT   STRING(RECORD) EDIT(NAME,PAY#,HOURS*RATE)
      (X(1),A(12),X(10),A(7),X(10),F(8));
WRITE  FILE(OUTPRT)  FROM(RECORD));
```

3. 모양 지정 (模様 指定 picture specification)

편집은 모양 지정 (picture specifcation) 으로도 할 수 있다. 이것이 본격적 방법이 된다고 보겠다. (모양 지정에 관한 상세한 논술은 제Ⅱ부, 제4장, 〃모양 지정 문자〃에 있다.)

모양 지정은 문자 줄 자료나 수치 문자 자료를 기술한다. 수치 문자 자료는 수치를 나타낸다. 모양 지정은 따옴표에 싸이며 DECLARE 문에서 PICTURE속성을 가지고 사용된다.

```
DECLARE CODE PICTURE 'XXXX',
        PAYMT PICTURE '$999V.99',
```

〔1〕 문자-줄 모양 지정(文字-줄 模様 指定 character-string picture specification)

문자-줄 모양 지정은 문자 줄을 기술한다; 지정에 있는 모양 문자 수는 줄의 길이를 표시한다. (단지 X모양 문자만이 문자-줄 모양 지정에 사용된다)

보 기:

```
CODE = 'AZB908'; CODE = '4F';
```

〔2〕 수치 문자 모양 지정(数値 模様 指定 numeric character picture specificotion)

모양 문자 9 이외에도, 수치 문자 지정 수치 문자 자료를 편집하기에 사용되는 모양 문자를 가질 수 있다.

문자 줄 변수에 대입하면 왼쪽부터 오른쪽으로 되고; 채우기나 자르기는(padding and truncation) 오른쪽에서 된다.
수치 문자 변수에 대입하면 소수점의 위치에 따라 좌우된다(소수점은 모양 문자 V로 지정된다)

〔3〕 수치 문자 변수의 값(Value)

수치 문자 변수의 값은 두가지로 해석될 수 있다. 산수값으로서 또는 문자 줄 값으로서이다.

만일 수치 문자가 9로서만 이루어져 있다면 숫자는 고른수이며 문자와 길이가 같다. 만일 수치 문자가 편집 문자 (editing

character)를 포함하고 있으면, 수치와 문자 줄은 다르다. 편집 문자는 정해진 자리에 실제로 기억된다.

그러므로 편집 문자는 변수의 문자 줄 값의 일부가 된다. 그러나 편집 문자는 변수의 산수 값의 일부가 아니다. 이 산수 값은 십진수, 소수점의 위치, 음양 부호(만일 하나만 있으면)로 된다.

만일 수치 문자의 값이 다른 수치 문자 변수 또는 규약 산수 변수에 대입되면, 산수 값만 대입된다. 규약 산수 변수에 대입함에 있어서 (또는 산수 식 연산에 있는 수치 문자 변수의 출현에서), 규약 산수로 변환이 이루어 진다. 규약 산수로 변환될 때, 단지 부호만 검사될 것이다.

만일 수치 문자 변수의 값이 문자 줄 변수에 대입되면, 변환이 없이 대입 된다.

본래의 문자 줄 변수 (CHARACTER 속성으로 지정된다)가 수치 문자 변수 위에 정의될 수 있다.

보 기:

```
DCL A PIC '$999V.99', B CHAR(7) DEFINED A,
    C DEC FIXED(5,2), D CHAR(7),
    E PIC '$ZZ9V.99';
A = 128.76; C = A; E = 456.88; D = E;
```

상수가 A에 대입된 뒤에, 이의 산수 값은 128.76이다. 이 값이 C에 대입된다. A의 문자 줄 값은 $128.76이다. E를 직접 D에 대입될 수 있다.

산수 변수(다른 수치 문자 변수를 제외하고는)는 오류가 되지 않고 수치 문자 변수 위에 정의될 수 없다.

〔 4 〕 편집에 이용되는 수치 문자 자료

```
DCL COUNT PIC'99999';
COUNT=123.45;
```

COUNT는 소수점을 가르키는 V가 없으므로 맨 오른편 다음에 있는 것으로 대행된다. 고로 결과는 '00123'이 되어 소수 이하가 잘린다.

```
DCL TOTAL PIC'999V99';
TOTAL=123;      TOTAL=123.00;
```

위 두 문은 등가이다. TOTAL에는 소수점 위치를 말해주는 V가 있다. 그래서 결과는 양자 모두 '12300'이 된다. 즉 V는 문자로 존재하지 않는다. 그러나 TOTAL을 인용하면 123.00이 인용되는 것이다.

```
DCL SUN PIC'999V.99';
```

이는 6자리의 PIC(.도 참가)이다.

```
SUM=123;  SUM=123.00;
```

위 두 문은 등가이다. 결과는 '123·00'이다.

```
DCL RATE PIC'9V99.99';
RATE=7.62;
```

위 문의 결과는 '762.00'이다. 그러나 이의 산수 값은 7.6200이다. 고로 편집 문자 .는 소수점 위치와는 관계 없다.

```
DCL RESULT PIC'9.999.999,V99';
RESULT=1234567;
```

RESULT의 문자 줄 값은 '1.234.567,00'이다. 이의 산수 값은 1234567.00이다.

```
DCL TALLY PIC'ZZ9';
TALLY=12;
```

TALLY의 문자 값은 'bb12'이다.

```
DCL PRICE PIC '$99V.99';
PRICE=12.45;
```

PRICE의 문자 줄 값은 '$12.45'이다.

이의 산수 값은 12.45이다.

그외 모양 지정은 부동점수와 영국 돈 형식을 지정할 수 있다. 이들은 제Ⅱ부, 제4장, "모양 지정 문자"에서 논술된다.

〔5〕 수치 문자 자료의 이용

수치 문자 모양 지정의 한가지 목적은 인쇄될 자료를 편집하는 것이다. 그러나 PL/I에서 이 목적에만 제한되는것은 아니다.

보 기 :

```
DCL RESULT PIC 'XXXXXX', COST PIC '$9V.99';
COST=7.15; RESULT=COST;
```

COST의 문자값은 '$7.15'이다. RESULT는 '$7.15b'이다.

```
DCL RESULT FIVED DEC(3,2),
    COST PIC '$9V.99';
COST=1.10; RESULT=2 * COST;
```

위에 보인 것처럼 수치 문자는 수로서의 성질도 갖는다.

4. 문자-줄과 비트-줄 내조립 함수(內組立 函数 bult-in function)

PL/I은 언어의 줄-취급 능력을 부과하기 위하여 많은 내조립 함수(이 중에 셋은 유사 변수(Pseudo-Variables)임)를 준비하고 있다. 다음은 이들 함수에 관한 대강의 설명이다(좀더 자세한 설명은 제Ⅱ부, 제7장, "내조립 함수와 유사-변수"에

있다)

STRING 내조립 함수는 UNALIGNED 구조체의 요소가 단일 문자줄로 이어질 것임을 지정한다. 전 요소는 문자 줄이거나 수치 문자난이어야 한다. 이는 유사 변수가 되기도 한다. 유사 변수로 작용하면 그 역이다.

BIT 내조립 함수는 자료 항목이 비트 줄로 변환될 것임을 지정한다. 이 내조립 함수는 프로그래머에게 자료 변환의 기본 규칙에서 도출되는 길이를 무시하고 변환되는 줄의 길이를 지정할 수 있게한다.

CHAR 내조립 함수는 변환이 지정된 길이의 문자 줄로 되는 것을 제외하고는 BIT 내조립 함수와 같다.

SUBSTR 내조립 함수는 지정된 줄 값으로부터 특정 버금줄 (substring)을 끄집어낼 수 있게 한다. 유사 변수로 작용할 때는 이 반대다. 이는 유사 변수가 되기도 한다.

INDEX 내조립 함수는 줄에서, 문자 줄이건 비트 줄이건, 지정된 버금줄의 첫번째로 나타나는 곳을 찾게할 수 있다.
돌아가는 값은 줄의 처음에서부터 센 그 버금줄의 첫번째 위치이다. 이 값은 이진 고른수이다. 만약 발견되지 않으면, 이 값은 0이다.

HIGH 내조립 함수는 대조 순서상의 최고위문자가 반복해서 들어감을 지정한다. 조직/360 실무에서, 이 문자는 10진수로 FF이다

LOW 내조립 함수는 대조 순서상의 최저위 문자가 반복해서 들어감을 지정한다. 조직/360 실무에서, 이 문자는 16진수로 00이다.

REPEAT 내조립 함수는 문자줄이 지정된 버금줄의 반복으로 이루어지게 한다.

BOOL 내조립 함수는 두개의 지정된 비트 줄에 16가지 BOOL 연산의 하나를 작용시킨다.

UNSPEC 내조립 함수는 어떤 값의 내부 규약 표현이 변환 없이 비트 줄로 바뀜을 지정한다. 유사 변수로 작용되면 그 반대다.

제 10 장 버금과정과 함수 (버금過程과 函数 subroutines and functions)

제 1 절 인수와 매개변수 (引数와 媒介変数 arguments and parameters)

자료는 그 자료를 불러내지는 수속 (invobed procedure) 에 들어가도록 지정하여 이름의 범위를 확장함으로써 불러내지는 수속에 알려지게 할 수 있다. 이 이름의 확장은 수속을 품음으로써 또는 이름을 위해 EXTERNAL 속성을 지정하므로써 소기의 목적을 달성할 수 있다.

또 다른 방법은 불러내는 수속에서 인수 (引数 arguments) 로 이름을 지정하는 것이다. 각 인수는 불러내지는 수속에 (invobed procedure) 보내진다.

인수가 보내지므로, 불러내는 수속은 이들을 받아드리는 어떤 방법이 있어야 한다. 이는 수속이 불러내지는 입구점 (entry point) 이 되는 PROCEDURE나 ENTRY 문에 있는 하나 이상의 매개변수 (媒介変数 parameter) 의 명시 선언으로써 이루어진다. 매개변수란 이 수속에 인수로써 보내어진 또 다른 이름들 (또는 식) 표현하기 위하여 불러내지는 수속 안에서 사용되는 이름이다. 불러내지는 수속의 매개변수 나열에 있는 각 매개변수는 불러내는 수속의 인수 나열에 있는 반응하는 인수를 갖는다. 이 대응은 왼편에서 오른편으로 취해진다; 첫째 인수는 첫째 매개변수에 대응하고, 두번째는 두번째로의 식이다. 인수의 수와 매개변수의 수는 동일해야 한다.

보 기 :

```
PRMAIN: PROCEDURE;
        DCL NAME CHAR(20), ITEM BIT(5), OUTSUB ENTRY;
        ...
        CALL OUTSUB (NAME, ITEM); ..
        END;
OUTSUB: PROC(A,B);
        DCL A CHAR(20); B BIT(5);
        ...
        PUT EDIT(A,B) (A(20), B(5)); ...
        END;
```

위 보기에서 NAME은 A, ITEM은 B와 대응한다. PRMAIN은 불러내는 수속, OUTSUB는 불러내지는 수속이다.

인수를 보낸다는 것은 이름을 보내는 것을 말하며 이들 이름에 의해서 표현되는 값을 보내는 것이 아님을 주목하라. (보통, 보내지는 이름은 값을 가진 주소이다.) 결과로서, 인수로서 보내지기 전에 변수에 할당된 기억소는 이 소속이 불러내질 때 중복되지 않는다. 매개변수로 지정된 값의 어떤 변동도 실제로 인수의 값을 변동시키는 것이다. 이런 변동은 통제가 불러내는 블러크로 돌아갈 때 효력을 가진다

매개변수는 인수로 직접 표현되는 값을 간접으로 표현하는 것이라 생각될 수 있다. 인수와 매개변수는 동일 값을 나타내므로, 매개변수와 이의 대응하는 인수의 속성은 같아야 한다. 보기로,만일 매개변수가 속성 FILE과 이에 대응하는 인수가 속성 FLOAT를 가지면 오류가 존재한다.

이름은 PROCEDURE나 ENTRY문의 매개변수 나열에 나타나므로서 매개변수로서 명시 선언된다. 그러나, 이의 속성은, 태만이 적용되지 않는한, 수속 안에 있는 DECLARE 문에서 명시 선언되어야 한다. ENTRY 문에서 지정된 매개변수는 품은 수속의 PROCEDURE 문에서나, 이 수속내의 DECLARE 문에서 지정되어야 한다.

매개변수는 일반화 수속에서 그런 수속에 알려져 있지 않은 이름의 자료가 연산되는 방법을 제공한다. PL/I에서 이런 일반화 수속은 두 종류가 있다 : 버금과정 수속 (간단히 버금과정이라함) 과 함수 수속 (함수)

제 2 절 버금과정 (subroutine)

버금과정이란 통상 불러내는 CALL 문에서 수속으로 보내질 인수를 요하는 수속이다. 이는 외부 (external) 또는 내부(internal) 수속이 될 수 있다. 이런 수속에 관한 인용 (引用 reference) 을 버금과정 인용 (버금過程 引用 subroutine reference) 이라 한다. 버금과정 인용의 일반 형식은 다음과 같다.

CALL 입구 - 이름 [(인수 [, 인수] . . .)] ;

버금과정이 불러내지면, 인수는 매개변수와 연관되고 버금과정은 기동되며, 실행이 시작한다. 버금과정이 마침에 따라, 통제는 정상으로 불러낸 블러크로 돌아간다. 버금과정은 정상으로 다음 방법의 하나로 끝날 수 있다.

① 통제가 버금과정의 마지막 END문에 닿는다. 이 문의 실행은 통제를 이 버금과정을 불러낸 논리상 다음의 첫번째 실행가능한 문으로 돌려보내게 한다. 이는 정상 복귀라

여겨진다.

② 통제가 버금과정 안의 RETURN 문에 닿는다. 이는 END 문에 의한 정상 복귀와 같다.

③ 통제가 통제를 버금과정 밖으로 옮기는 GO TO 문에 닿는다 이 GO TO 문은 버금과정에 알려진 포함한 블러크에 있는 명찰을 지정하거나, 버금과정에 보내진 명찰 인수와 연관된 매개변수를 지정해야 한다. 이는 버금과정의 정상 마침이라 여겨지나, 이는 END나 RETURN문에 의한 효과처럼 통제의 정상 복귀는 아니다.

버금과정에서 STOP 문은 이 버금과정과 이를 불러낸 수속과 연관된 전 프로그램의 실행을 끝낸다.

보 기 :

```
PRMAIN: PROC OPTIONS (MAIN);
          . . .
A: PROC;
   DCL RATE FIXED(10,3), TIME FIXED(5,2), DISTANCE
   FIXED(15,5), MASTER FILE . . . ;
   . . .
   CALL READCM(RATE, TIME, DISTANCE,MASTER );
   . . .
  END;
READCM: PROC(W,X,Y,Z
        DCL W FIXED (10,3),X FIXED(5,2),
        Y FIXED(15,5),Z FILE . . . ;
        . . .
```

```
GET FILE(Z) EDIT(W,X,Y)F(10,3),F(5,2),F(15,5);
. . .
y = W * X ;
IF y > 0  THEN RETURN;
ELSE PUT EDIT( 'ERROR READCM')(A(12));
END; . . .
END;
```

제 3절 함수(函数 function)

함수(function)란 이것이 불려내질 때 통상 이것으로 보내지는 인수를 요하는 수속을 말한다. 버금과정과 달라서, 함수는 식에 있는 함수 이름(그리고 연관된 인수)의 출현에 의해서 불려내진다. 그런 출현을 함수 인용(function reference)이라 한다. 버금과정 처럼, 함수는 이것에 보내진 인수 그리고 다른 알려진 자료에 연산될 수 있다. 그러나 버금과정과 달라서, 함수는 함수 인용점으로 통제와 함께 돌아오는 단일 값을 계산하기 위해서 쓰여진다. 이 단일 값은 산수, 줄, 모양, 지침 값이 될 수 있다.

보 기 :

```
MAINP:PROC;
      DCL SPROD ENTRY; . . .
      X=Y * * 3 + SPROD(A,B,C) ; . . .
      END;
```

위에서, 대입 문

```
X = Y * * 3 + SPROD(A,B,C) ;
```

는 SPROD라 하는 함수에 관한 인용을 포함한다. 이 SPROD는

다음과 같이 되어 있다고 한다.

```
SPROD: PROC (U,V,W) ;
        . . .
        IF U > V + W  THEN RETURN(Ø) ;
       ELSE RETURN (U * V * W ) ; . . .
       END;
```

여기서 A, B, C와 U, V, W는 태만 속성으로 다같이 FLOAT DECIMAL(6)를 가지게 된다. 고로 속석이 일치한다. 그러므로 오류가 없다.

RETURN 문은 함수를 끝마치게 하고 통제를 불러내는 수속에 돌려보내는 통상의 방법이다. 이의 사용은 버금과정에서의 사용과는 좀 다르다. 이는 통제뿐 아니라 하나의 값을 돌려보낸다.

함수에서 사용되는 RETURN 문의 형식은 다음과 같다.

```
RETURN ( 요소-식 ) ;
```

이 식은 꼭 나타나야 하고 단일 값을 표현해야 한다. 이는 배열이나 구조체 식이 되지 못한다. 이 값이 불러내는 수속에 있는 불러낸 점으로 돌아간다. 함수 인용은 이 값으로 대체된다. 그리고 값내기가 계속한다.

함수는 또 GO TO 문의 실행으로 끝마칠 수 있다. 만일 이 방법이 사용되면, 함수를 불러낸 식의 값내기는 마치지 못하고, 통제는 지정된 문으로 갈 것이다. 버금과정에서, GO TO 문에서 정된 이행 점(移行 点)은 명찰 인수와 연관된 매개변수가 될 수 있다.

보 기 :

```
MAINP: PROC;
```

```
DCL SPROD ENTRY ; . . .
X = Y ** 3 + SPROD(A,B,C,LAB1);
...
LAB1: CALL ERRT ; . . .
END;
SPROD: PROC(U,V,W,Z);
  DCL Z LABEL ; . . .
  IF U > V + W  THEN GO TO Z;
  ELSE  RETURN(U * V * W ) ;
  . . .
  END ;
```

위에서 GO TO 문이 실행된다면, SPROD가 들어있는 식 X = Y ** 3 + SPROD는 값내기가 중단되고 LAB1이 붙은 문으로 간다.

주 : 어떤 경우는, 함수는 인수를 요하지 않게 정의될 수 있다. 이 경우, 식안에 함수 이름의 출현은 만일 함수 이름이 블록 어느곳에 있는 입구 이름으로서 선언되었을 때만 함수 인용으로서 인지될 것이다. 자세한 지식을 위하여는 이 장의 "ENTRY 속성"을 보라.

함수 인용에 의해 기동된 수속이 RETURN (식) 이나 GO TO 어느것에 의해서도 끝나지 않을 때, 결과는 알 수 없다.

〔 1 〕 함수에 의해 돌아간 값의 속성

함수에 의해 돌아간 값의 속성은 두가지로 선언될 수 있다.

① 이들은 함수 이름의 첫 문자에 따라서 태만에 의해 선언된다.

② 이들은 함수 PROCEDURE (또는 ENTRY) 문에 있는 매개변수 나열 다음에 선언된다.

어느 방법이 채택되든 간에, 이차 입구점을 위한 자료 속성은, 어떤 태만 속성에 의하던, 일차 입구점에서 설정된 속성과 같아야 한다. 달리 말하면, ENTRY 문에서 지정된 속성은 (명시로나 태만으로나) PROCEDURE 문에서 지정된것과 같아야 한다.

RETURN 문에 있는 식의 값은, 필요하다면, 위에서 말한 두가지 중의 하나로 지정된 속성에 일치하게끔 함수 내에서 변환된다.

앞 보기에서, SPROD의 PROCEDURE 문은 돌아가는 값을 위해서 선언된 속성을 가지고 있지 않다. 그러므로, 이 속성은 이름의 첫 문자, S에 의해 결정된다. 이는 FLOAT DECIMAL(6)이다.

다음 경우는 속성을 지정해 주는 경우의 보기이다.

```
SPROD: PROC(U,V,W,Z) FIXED BINARY;
```

이 경우, RETURNS 속성이 이 함수를 불러낸 수속 안에서 함수 입구 이름을 위한 DECLARE 문에서 지정되어야 한다.

태만 속성이 입구 이름에 적용되지 않는한, 어떤 함수의 불러내기도 그 입구 이름을 위한 RETURNS 속성 선언의 범위 안에서 나타나야 한다. 내부 함수에서, RETURNS 속성은 함수 수속과 동일하게 내적인 DECLARE 문에만 지정될 수 있다. 만일 RETURNS 속성이 내부 함수에 선언되면, INTERNAL 속성이 동일 선언에서 지정되어야 한다. RETURNS 속성의 일반 형식은 다음과 같다.

```
RETURNS ( 속성 - 나열 )
```

이 속성은 함수에서 돌아오는 속성과 같아야 하며, 만약 다르면 오류이다. 앞서 보기와 같이 선언되었다면, 다음선언이 MAINP

안에서 주어져야 한다.

```
DECLARE SPROD RETURNS(FIXED BINARY);
```

〔 2 〕 내조립 함수 (內組立 函数 builtin function)

프로그래머 자신이 정의할 수 있는 함수 수속과 비슷하게 내조립 함수 (builtin function)라 하는 미리 정의해놓은 편리한 함수의 집합이 있다. 내조립 함수의 집합은 PL/I의 고유한 일부이다. 이는 흔히 사용되는 산수 함수만이 아니고 줄과 배열 취급하는 함수와 같이 유용한 언어 설비에 관계되는 함수도 포함하고 있다. 내조립 함수는 프로그래머가 정의한 함수가 불러내지는 것과 같은 방법으로 불러내진다. 그러나 많은 내조립 함수가 배열 값을 돌려보낸다.

ENTRY나 RETURNS 속성은 지정되지 못하며, 내조립 함수에 의해 돌아가는 값의 속성은 편성자에 알려져 있다.

내조립 함수 이름은 PL/I 열쇠말이므로, 이들은 예약된 것이 아니다; 동일한 표식어가 프로그래머가 정의한 이름으로 사용될 수 있다. 만일 내조립 함수 (또는 유사 - 변수) 인용이 동일 표식어가 다른 목적으로 선언된 다른 블러크에 포함된 블러크에서 사용되면 모호한 것이 될 것이다. 이 모호성을 피하기 위하여, BUILTIN 속성이 물려준 어떤 블러크에 있는 함수 이름에 선언될 수 있다.

보 기 :

```
A : PROC ;
    . . .
  B : BEGIN ;
      DCL SQRT FLOAT BINARY ;
      . . .
```

```
C : BEGIN ;
    DCL SQRT BUILTIN ; . . .
    END ;
. . .  END ;
. . .  END ;
. . .  END ;
```

외부 수속 A에서는 SQRT가 다른 목적으로 명시 선언되지 않았다고 가정한다. 그러면, SQRT에 대한 어떤 입구 이름 인용도 내조립 함수로서 문맥상 선언이 될 것이다. B에서는, SQRT는 부동점 이진수 변수로 선언되었고, 이는 어떤 다른 목적으로도 사용못된다. 다시 C에서, SQRT는 SQRT에 관한 어떤 인용도 내조립 함수로서 인지될 것이고 B에서 선언된 고정점 이진수 변수로 인지 안되게 BUILTIN 속성으로 선언되었다.

보 기 :

```
SQRT: PROC(PARAM) FIXED(6,2);
      DCL PARAM FIXED(12);
      . . .
      END ;
A : PROC ;
    DCL SQRT ENTRY RETURNS(FIXEI)(6.2)),Y FIXED(12);
    . . .
    X = SQRT(Y);
    . . .
    B : BEGIN ;
    DCL SQRT BUILTIN ;
    . . .
```

```
        Z = SQRT(P);
        . . .
        END; . . .
    END ;
```

이 경우, SQRT는 A에서 내조립 함수가 아니므로, 입구 이름이 명시 선언되어야 한다. 블러크 B에서는 SQRT는 내조립 함수이다.

만일 내조립 함수의 이름을 사용한 프로그래머가 쓴 함수가 내적이면, 인용되는 블러크에 이것의 이름이 알려지는 동안은 이 표식어에 대한 어떤 인용도 프로그래머가 쓴 함수에 관한 인용도 프로그래머가 쓴 함수에 관한 인용이 될 것이다.

제 4 절 ENTRY 속성

이미 언급한 바와 같이, ENTRY 속성은 연관된 표식어가 입구 이름임을 가르키는데 사용된다. 이런 지적은 만일 표식어가 다른 방법으로 입구 이름으로서 인지되지 못한다면 필요하게 된다. 다른 방법이란 PROCEDURE나 ENTRY 문의 명찰로서 나타나므로 인해 입구 이름으로 명시 선언되는 것이다.

그러므로, 만일 인용이 이것이 나타나지 않은 블러크에서 입구 이름에 관해 수행된다면, 이 표식어는 이 블러크 안에서 명시로나 암시로나 ENTRY 속성을 받아야 한다.

보 기 :

```
    A : PROC ;
        . . .
```

```
    PUT EDIT(RANDOM) (E(10,5));
    . . .
    END ;
```

위에서 A가 외부 수속이고 RANDOM이 인수를 요하지 않은 외부 함수라 하자. 이것으로는 RANDOM이 입구 이름으로서 인지되지 못한다. 그리고 PUT 문의 결과는 부정(不定)이다. RANDOM이 A 안에서 입구 이름으로서 인지되기 위하여는, 이는 ENTRY 속성을 가지고 선언되어야 한다.

보 기 :

```
A : PROC ;
    DCL RANDOM ENTRY;
    . . .
    PUT EDIT(RANDOM) (E(10,5));
    . . .
    END;
```

주 : ENTRY 속성은 암시에 의해서 명시 선언될 수 있다.
RETURNS 속성을 가지고 명시 선언된 어떤 표식어도 암시에 의해서 ENTRY 속성을 받는다.
고로, RETURNS는 ENTRY를 암시한다.

인수로서의 입구 이름

함수 또는 버금과정 인용의 인수는 인수 자체가 입구 이름이 될 수 있다.

이런때 다음의 하나에 속한다.

① 만약 입구 이름 인수가, 이를 FUNC라 하자, 자신의 인수

나열을 가지고 지정되면, 이는 함수 인용으로서 인지된다; FUNC가 불러내지고, FUNC에 의해 돌아온 값이 이것과 대치된다.

② 만일 입구 이름 인수가 인수 나열 없이 나타나고. 그리고 연산 식 안에나 괄호에 싸여 나타나면, 이는 인수 없는 함수 인용으로 간주된다. 보기로:

```
CALL A ((B));
```

는, 여기서 B는 입구 이름이다. A의 인수이며 이 함수 수속 B에 의해 돌아가는 값을 보낸다.

③ 만일 입구 이름 인수가 인수 없이 나타나며 연산 식에도 괄호에도 아니면, 이 입구 이름 자체가 항상 불러내질 함수나 버금과정으로 보내진다. 이 경우, 입구 이름은 함수 인용으로서 해석되지 않는다. 만일 이것이 인수를 요하지 않는 함수 이름일 때라도 그렇다. 보기로

```
CALL A (B);
```

는, 여기서 B는 입구 이름이 다, 입구 이름 B를 A의 인수로서 보낸다.

보 기 :

```
CALLP: PROC;
       DCL (RREAD, SUBR, ASQRT) ENTRY;
       . . .
       GET EDIT(R,S)(2F(10,5)); . . .
       CALL SUBR(RREAD,ASQRT(R),S,LAB1);
       . . .
```

```
LAB1: CALL ERRT(S);
        . . .
      END;
SUBR: PROC(NAME, X, Y, TRANPT);
      DCL NAME ENTRY, TRANPT LAB1;
      . . .
      IF X>Y THEN CALL NAME(Y);
      ELSE GO TO TRANPT;
      . . .
      END;
```

이 보기에서, CALLP,SUBR,ASQRT,RREAD는 외부 수속이라 하자. CALLP에서, 이름 RREAD,SUBR,SQRT는 ENTRY 속성을 가지고 명시 선언되었다. 4개 인수가 CALL SUBR 문에서 지정된다. 이들은 다음과 같이 해석된다.

① 첫째, 인수 RREAD는 입구 이름으로 인지되었다. 이는 자신의 인수 나열을 갖고 있지 않고, 연산 식 또는 괄호에 싸여 나타나 있지 않으므로, 입구 이름은 자체가 불러내지기 위해 보내진다.

② 둘째 인수 ASQRT(R)는 입구 이름으로 인지되어 있다. SQRT는 인수 나열을 가지므로, 이는 함수 인용으로 인지된다. ASQRT가 불러내지고 ASQRT에 의해 돌아오는 값은 가인수(仮引数 dumny argument)에 대입되고(〃가인수〃를 보라), SUBR가 불러내질 때, 이 가인수가 이에 보내진다.

③ 셋째 인수 S는 그대로 보내지는 단순한 십진 부동점수

요소 변수이다.

④ 넷째 인수 LAB1은 문 명찰 상수이다. 상수이므로, 가인수가 이를 위해 만들어진다. SUBR가 불러내질 때, 이 가인수가 보내진다.

제5절 인수와 매개변수의 관계

함수나 배금과정이 불러내질 때, 하나의 관계가 불러내는 문 또는 식의 인수와 불러내지는 입구점의 매개 변수 사이에 설정된다 이 관계는 가인수가 만들어지느냐에 좌우된다.

1. 가인수(仮引数 dummy argument)

앞에서 인수의 값이 아니고 인수의 이름이 버금과정이나 함수에 보내진다고 했다. 그러나, 인수가 이름을 갖지 않는 때가 있다. 보기로,상수는 이름이 없다. 또 연산 식도 이름이 없다. 그러나 인수를 매개변수에 연관시키는 설비는 그런 값을 직접 취급하지 못한다. 고로, 편성자는 그런 값을 위해 기억소를 준비해야 하고 이를 위해 내부 이름을 배정해야 한다. 이들 내부 이름을 가인수(仮仮引数 dummy argument)라 한다. 이들은 PL/I 프로그래머와 무관하나,매개변수에 관한 어떤 변동도 가인수의 값에만 영향을 주고 원래의 인수 값에는 영향을 안 주기때문에 이들 가인수의 존재에 관심을 가져야 한다.

가인수는 다음의 어떤 경우이면 항상 만들어진다.

① 인수가 상수, 보기로, CALL X(6.25);

② 인수가 연산자를 가진 식, 보기로, CALL(A+B); CALL(+A);

③ 인수가 자체로서 인수를 가진 함수 인용, 보기로, CALL X(SIN(Y));

④ 인수가 괄호에 싸인 식, 보기로, CALL X(A)); 이 보기에서 보인 것처럼, 인수는 이것이 수속에 보내지고, 인수의 값이 불러내지는 수속에 의해 변동되지 않을 경우면 괄호에 싸여도 상관 없다.

⑤ 인수가 명찰 상수.

다른 모든 경우에는, 인수 이름이 직접 보내진다. 매개 변수는 보내진 인수와 같다; 그래서, 매개변수 값의 변동은 만일 가인수가 아닌 경우라면 원래의 인수 값에 영향을 준다.

2. 인수와 매개변수 형식

보통, 인수와 이에 맞서는 매개변수는 어떤 자료 구성과 형식도 된다. 보기로, 인수는 맞서는 매개변수가 지침으로 마련되었다면 지침이 된다; 맞서는 매개변수가 비트 줄이면 인수는 비트 줄이 된다. 그러나, 모든 매개변수/인수 관계가 그렇게 명백하지만은 않다. 어떤 것은 더 자세한 정의와 분명성(分明性)을 필요로 한다. 이런 경우가 아래에 주어진다.

만일 매개변수가 요소이면, 이는 구조체도 배열도 아닌 변수를 이름, 인수는 요소 식이어야 한다. 만일 인수가 첨자 붙은 변수이면, 첨자는 버금과정이나 함수가 불러내지고 지정된 요소의 이름이 보내지기에 앞서 값이 내진다.

만일 매개변수가 배열이면, 인수는 배열 이름이어야 한다. 인수의 자료 속성은 매개변수의 그것에 맞아야 한다.

배열 인수의 한계는 배열 매개변수의 그것에 맞아야 한다.

만일 매개변수가 구조체이면, 인수는 구조체 이름이어야 한다. 인수와 매개변수의 상대적 구조는 같아야 한다; 수준 번호가 같을 필요는 없다. 구조체 인수의 요소 자료 속성과 맞서는 매개변수의 요소의 그것과 맞아야 한다.

만일 매개변수가 요소 명찰 변수이면, 인수는 요소 명찰 변수이거나 명찰 상수이어야 한다. 만일 인수가 명찰 상수이면, 가인수가 만들어진다.

만일 매개변수가 배열 명찰 변수이면, 인수는 동일 한계를 가진 배열 명찰 변수이어야 한다.

만일 매개변수가 입구 이름이면, 인수는 입구 이름이어야 한다. 내조립 함수의 이름은 보내지지 못한다. (그러나, 내조립 함수 인용은 함수 이름이 아니라 함수 인용의 값이 보내지기 때문에 인수 나열에 나타날 수 있다.)

만일 매개변수가 화일 이름이면, 인수는 화일 이름이어야 한다. 보통, 화일 이름 인수의 속성은 화일 이름 매개 변수의 그것과 일치해야 한다. 그러나, D-편성자에서, 어떤 경우는, 같을 것이 요구되지 않는다. 이것은 BACKWARDS 속성과 다음에 있는 ENVIRONMENT 속성의 선택항에만 맞는다.

BUFFERS(n), CTL360, LEAVE, INDEXMULTIDLE, NOLABEL, HIGHINDEX, VERIFY, OFLTRACKS, MEDIUM, INDEXAREA, CTLASA, ADDBUF

MEDIUM 선택항의 경우에는, 논리장치 이름만이 다를 수 있다; 물리 장치 형식은 같아야 한다.

화일 이름 인수가 위의 어떤 경우에서 매개변수와 같지 않으면 인수의 것이 공급되고 매개변수의 맞지 않은 FNVIRONMENT 속성과

BACKWORDS 속성은 무시된다. 다른 모든 경우, 맞을 것이 항상 요구되고 만일 어떤 속성이 맞지 않으면 오류이다.

보 기 :

```
A : PROC OPTIONS(MAIN);
    DCL X FILE RECORD INPUT BACKWARDS ENV(FI8Ø) MEDIUM
    (SYSØØ1, 24ØØ) BUFFERS(1) LEAVE),Y FILE RECORD
    INPUT ENV(FI8Ø) MEDIUM(SYSØØ2, 24ØØ) BUFFERS(2));
    . . .
    CALL B(X) ; . . .
    CALL B(Y) ; . . .
    . . .
B : PROC(Z);
    DCL Z FILE RECORD INPUT ENV(F18Ø) MEDIUM (SYSØØØ,
    24ØØ));
    . . .
    OPEN FILE(Z);
    . . .
    END B ; . . .
    END A;
```

이 보기에서, X는BACKWARDS 속성을 가지나 이에 맞서는 매개변수는 갖지 않는다. 이는 위에서 주어진 경우의 하나이므로, 같을 것이 요구 안되며 Z는 첫번 B가 불려내질 때 BACKWARDS 속성을 받는다. 유사하게, 논리 장치 이름 SYØØ1, BUFFERS(1), LEAVE 선택항을 받는다. 그러므로 OPEN 문은 X의 모든 속성을 가지고 Z를 여는 것이 된다.

두번째는 역시 Y의 모든 속성과 선택항을 Z가 받는다. X와는 관련이 없다.

만일 매개변수가 요소 지침 변수이면, 인수는 요소 지침 변수이거나 요소지침 식이어야 한다.

만일 매개변수가 지침 배열이면, 인수는 동일한계를 가진 지침 배열이어야 한다.

만일 매개변수가 배열 또는 줄이면,배열의 한계 또는 줄의 길이는 이들이 매개변수 아닌 것에 지정된 거와 같은 방식으로 지정되어야 한다. 이는 십진 고른수 상수등의 방식처럼. 이들은 맞서는 인수와 한계와 길이가 같아야 한다.

만일 식이 인수로서 보내지면, 기수, 척도, 정도 - 이는 정도 규칙에의 정해지는 결과의 그것 - 은 맞서는 매개변수에 지정된 것과 같아야 한다. 또한, 인수로서 보내지는 산수 상수의 기수, 척도, 정도는 이에 맞서는 매개변수의 그것과 같아야 한다. 비슷하게, 인수로서 보내지는 줄 상수의 길이는 이에 맞서는 매개변수의 그것과 같아야 한다.

매개변수는 기억소 종류를 갖지 않으므로 어떤 기억소 종류 속성을 가지고도 선언 못된다.

완전한 진단이 CALL과 PROCEDURE 문의 인수와 매개변수 나열에 대해 주어지지 않으므로, 모든 가능한 인수(또 매개변수)와 이들의 속성이 아래에 요약되었다.

CALL 입구-이름[(인수1[,인수2,...인수n])];

인수: 요소 식, 내조립 함수 이름, 배열 이름, 구조체 이름, 소구조체 이름, 화일 이름; 정해진, 정적(靜的), 자동적, 외부의, 매개변수;

입구, 명찰, 지침, 문자 줄, 비트 줄, 산수(이진, 십진, 고정점,

부동점, 수치 문자)

입구 - 이름 : PROCEDURE

〔 (매개변수 1 〔 , 매개변수 2 , . . . , 매개변수 n 〕) 〕

〔 OPTIONS (선택항 - 나열) 〕

{ 〔 RETURNS (자료 속성) 〕 | 〔 자료 속성 〕 } ;

매개변수 : 변수 이름, 배열 이름, 구조체 이름, 소구조체 이름, 화일 이름 ; 기억소 종류 없음 ;

입구, 명찰, 지침, 문자 줄, 비트 줄, 산수 (이진, 십진, 고정점, 부동점, 수치 문자)

제 11 장 예외 조건 취급과 프로그램 검토 (例外 条件 取扱과 프로그램 検討 exceptional condition handling and program checkout)

PL/I 프로그램이 실행될 때, 많은 수의 예외 조건이 조직에 의해 감시되고 이들의 출현은 자동으로 탐지된다. 이런 예외 조건은 넘치기 또는 입력/출력 전송 오류와 같은 오류가 되기도 하고, 또는 이들은 화일의 끝 또는 지면의 끝과 같은 예상은 되나 빈번하지는 않는 조건이 되기도 한다.

시험을 해볼 수 있는 각 조건은 이름이 부여되고, 이들 이름은 프로그래머에 의해 예외 조건을 취급하기 위하여 사용된다. 조건 이름은 PL/I 언어의 일부이다. 열쇠말 이름과 각 조건에 대한 기술은, 제Ⅱ부, 제 8 장, " ON 조건 "을 보라

제 1 절 가능하게 된 조건과 설정된 조건 (可能하게 된 条件과 設定된 行為 enabled conditions and established action)

감시되며, 이것의 출현은 중단 (中断 interrupt)을 가져오는 조건을 가능하게 되어 있다고 한다. 조건의 출현이 중단을 가져올 때 일어나게끔 지정된 행위를 설정되었다 (設定되었다 established)고 말한다. 대부분의 조건은 자동으로 검사되고, 그들이 일어나면, 조직은 통제를 발휘하고 그 조건을 위해 지정된 어떤 표준 행위를 수행한다. 이들 조건은 태만에 의해 가능하게 된다. 그리고 표준 조직 행위가 그들을 위해 설정된다.

대부분의 조직 행위는 ERROR 조건을 일으킨다. 이것은 각각의 오류 형식을 개별로 검사하기 보다는 여러가지의 오류를 검사하는데 사용된다. ERROR 조건에 대한 표준 조직행위는 프로그램을 끝나게 한다.

프로그래머는 어떤 조건이 가능하게 되어 있게끔 또는 그렇지 않게 지정할 수 있다, 이 말은 그것이 일어났을때 중단이 되게끔 검사된다는 말이다.

만약 조건이 불능하게 되어있으면(不能하게 되어 있으면 disabled), 조건의 출현은 중단을 초래치 않는다.

모든 입력/출력 조건과 ERROR 조건은 언제나 가능하고 불능하게 되지 못한다. 모든 계산 조건(計算 条件 computatond condition)은 가능하게 또는 불능하게 될 수 있다. SIZE 조건은 중단을 초래 하려면 명시로 가능하게 되어야 한다; 다른 모든 조건은 태만에 의해 가능하게 되고 만일 이들이 출현할 때 중단을 초래케 하지않으려면 명시로 불능하게 해야 한다.

〔 1 〕 조건 전치어 (条件 前置語 condition prefixed)

불능하다 가능하다 하는 것은 조건 전치어 (condition prefix)에 의해 어떤 조건을 위해 지정될 수 있다. 조건 전치어는 괄호에 싸이고 쉼표로 분리되며, 콜론(:)에 의해 문(또는 문 명찰)에 이어진 하나 이상의 조건 나열이다. 이 전치어는 문과 어떤 문 명찰에도 앞서야 한다. 전치어 나열에 있는 조건 이름은 당해 조건이 전치어의 범위(scope)안에서 가능하게 됨을 가르킨다. 어떤 전치어 이름은 해당 조건이 불능하게 됨을 가르키기 위하여 NO 자(字)를 앞에 붙일 수 있다.

〔2〕 조건 전치어의 범위 (scope 範囲)

전치어의 범위는, 이는 이것이 적용되는 프로그램의 부분이다. 보통 전치어가 부착된 문이다. 조건 전치어는 문의 실행에서 불러내지는 함수나 버금과정에는 적용되지 않는다.

IF 문에 붙은 조건 전치어는 IF 다음에 오는 식의 값내기에만 적용된다; 이는 THEN이나 ELSE 절에는 적용되지 않는다. 그러나 이들 절은 전치를 가질 수 있다. ON 문에 붙은 전치어는 연관된 ON-단위에 효력을 미치지 못한다. DO 문에 붙은 조건 전치어는 DO문 자체의 식의 값내기에만 적용되고 DO 모임 안에 있는 어떤 다른 문에도 적용되지 않는다.

PROCEDURE 문과 BEGIN 문에 붙은 조건 전치어는 특별한 경우 (자주 사용되지만)이다. PROCEDURE나 BEGIN 문에 부착된 조건 전치어는 END 문까지의 모든 문에 적용된다. 이는 그 블러크 안에 품어진(nested) PROCEDURE나 BEGIN 문도 포함시킨다. 이는 프로그램의 실행중 불러내지는 이 블러크의 밖에 있는 어떤 수속에도 적용되지 않는다.

조건의 가능하게 하기와 불능하게 하기는 블러크 안의 문에, 여기에는 PROCEDURE와 BEGIN문도 포함한다 (이것은 품어진 블러크 안에 있는 조건의 가능하게 하기와 불능하게 하기를 재정의하는 것이 된다), 부착함으로써 블러크 안에서 재정의(再定義)될 수 있다. 이런 재정의는 전치어가 부착된 문의 실행에만 정용된다. 품어진 PROCEDURE나 BEGIN 문의 경우에는, 이는 그 블러크안에 들어있는 블러크를 포함해서, 문이 정의한 블러크에만 적용된다. 통제가 재정의한 전치어의 범위 밖으로 가면, 재정의는 더이상 적용되지 않는다.

조건 전치어는 DECLARE나 ENTRY 문을 뺀 어떤 문에도 부착시킬 수 있다.

보 기 :

```
(SIZE) : A : PROC ;
         . . .
(NOSIZE) : B : BEGIN ;
         . . .
         END B ; . . .
         END A ;
```

이 보기에서, 조건 전치어 SIZE는 수속 A를 위해 이 조건을 가능하게 만들었다. 이 전치어는 블러크 B에도 적용될 것이나 NOSIZE를 B에 붙여 여기서는 SIZE를 불능하게 만들었다.

〔 3 〕 ON 문

조직 행위는 모든 조건에 존재한다. 그리고 만일 중단이 일어나면, 이 조직행위가 프로그래머가 그 조건에 대한 ON 문에서 별도의 행위를 지정하지 않으면 수행될 것이다. ON문의 목적은 태만에 의하거나 조건 전치어에 의하거나 가능하게 된 예외 조건때문에 중단이 초래될 때 취해지도록 행위를 설정하는 것이다.

주 : ON문에서 지정된 행위는 조건이 불능한 프로그램 부분에서는 실행이 안된다.

ON문의 형식은 :

```
ON 조건-이름 { SYSTEM;| on-단위 }
```

(제Ⅱ부, 제10장, 〃문〃을 보라).

열쇠말 SYSTEM은 중단이 일어났을 때 표준 조직 행위를 지정하는 것이다. on-단위는 중단 시 취해질 별도의 행위를 프로그래머에 의해 지정되는데 사용된다. on-단위는 빈 문(null slatement)이나 GO TO 문이 되어야 한다; 이는 명찰이 붙을 수 없다.

빈 문 on-단위는 중단을 무시한다. 그리고, 보통, 통제를 중단이 일어난 점 다음의 점으로 돌려보낸다. 그러므로, 빈 on-단위의 효과는 다음과 같이 말할 수 있겠다. 즉 〃이 조건의 결과로서 중단이 일어날 때, 그대로 계속한다.〃

빈 on-단위의 사용은 두가지 이유에서 조건의 불능하게 만들기와는 같지 않다. 첫째, 빈 on-단위는 ENDFILE, KEY, CONVERSION을 제하고는 어떤 조건에도 지정될 수 있다. 그러나 모든 조건이 다 불능하게 되지는 못한다; 둘째, 만일 가능하다면, 조건을 불능하게 만드는 것은 이 조건에 대한 어떤 검사도 그만둠으로써 시간을 절약할 수 있다.

만일 빈 on-단위가 지정되면, 조직은 계속 조건의 출현을 검사하고, 중단이 일어날 때 통제를 on-단위로 보내고, 그런 다음에 아무일도 않고, on-단위로부터 돌아온다.

주: 통제가 on-단위로부터 돌아오는 특정점은 조건에 따라 변한다. 대부분, 이는 조건이 일어난 행위의 바로 다음 곳으로 돌아간다. 제8장(제Ⅱ부), 〃ON-조건〃에서 빈 on-단위가 지정될 수 있는 조건에 대한 복귀점을 보여준다. 빈 on-단위로부터 돌아가는 것을 정상복귀(正常 復帰 normal return)라 한다.

만일 on - 단위가 GO TO 문이면, 그리고 중단이 일어나면, 통제는 GO TO 문에서 지정된 명찰로 옮겨간다. 중단이 일어난 점과의 연락은 끊어지고 정상 복귀는 일어날 수 없다.

〔 4 〕 ON 문의 범위

ON문의 실행은 행위 지정을 호명된 조건과 연관시켜 준다. 한번 이 관계가 설정되면, 이는 이것이 말소되거나 ON문이 실행되는 블러크가 끝날 때까지 지속한다.

설정된 중단 행위는 한 블러크로부터 이것이 기동시키는 어떤 블러크로도 간다, 그리고 이 행위는 동일 조건에 대한 다른 ON 문의 실행에 의해서 이것이 말소되지 않는한 다음에 오는 모든 기동된 블러크에 효과있게 지속한다. 만일 이것이 없어지면, 새로운 행위가 그 블러크가 끝날 때까지만 지속한다. 통제가 기동시킨 블러크에 돌아올 때, 그 점에 존재하던 모든 설정되었던 행위가 재설정된다. 이는 버금과정에 대해서 버금과정을 불러낸 블러크를 위해 설정된 중단 행위를 변경시키는것이 가능하다.

만일 동일 조건에 대한 둘 이상의 ON 문이 동일 블러크에 나타나면, 뒤에 오는 ON 문은 앞에 오는 ON 문을 무효로 만든다.

보 기 :

```
A : PROCEDURE ;
    DECLARE B ENTRY ;
    ON CONVERSION GO TO AERR ;
    ON ZERODIVIDE GO TO BERR ;
    . . .
    CALL B ;
    . . .
    END;
```

```
(NOOVERFLOW) : B : PROC ;
    DCL Z BIT(1), X CHAR(1);
    . . .
    ON CONVERSION GO TO CERR ;
    . . .
(NOCONVERSION) : Z = X ;
    . . .
    RETURN ;
    END ;
```

수속 A에 있는 ON 문은 A에서 일어나는 CONVERSION과 ZERODIVIDE 오류에 취해질 행위를 설정한다. (CONVERSION과 ZERODIVIDE는 태만에 의해 가능하게 되었고 고로 그것들을 가능하게 하기 위해서 조건 전치어를 요하지 않는다.) 이들 행위 지정은 수속 B로 옮겨간다. 이것은 A에 의해서 불러내지기 때문이다. 그리고 이는 B에 있는 ON 문 앞에까지 효과 있게 남아 있다. B의 ON 문은 CONVERSION 조건을 위한 새로운 행위를 설정한다. 이 새로운 행위는 B의 나머지 부분에서 효력을 가진다. 통제가 A로 돌아갈 때, A의 행위 지정이 재설정된다. (ZERODIVIDE를 위한 행위 지정은 B에서 변하지 않았으므로, 재설정될 필요가 없다).

B의 ON 문의 범위는 대입 문에는(Z = X ;)미치지 못한다. 그것은 NOCONVERSION 전치어가 그 문에 대한 CONVERSION조건을 불능하게 하기 때문이다. 그래서, 대입 문 실행 중의 CONVERSION 오류는 중단이 되지 않는다.

만약 CONVERSION 오류가 B의 ON 문이 실행되기에 앞서 일어나면, A에서 설정된 행위가 취해진다. 이것은 통제가 AERR로 옮겨감을 말한다. 같은 방법으로, B 안 어디서든 일어나는 ZERODIVIDE 오류는 BERR로 옮겨가는 결과가 된다.

OVERFLOW 조건은 태만에 의해 가능하게 되며, 그리고 A에 이에 관한 ON 문이 없으므로, A에서 OVERFLOW 오류는 OVERFLOW에 대한 표준 조직 행위를 취하게 한다. 그러나 B에서, OVERFLOW는 불능하게 되어 있다. 통제가 A로 돌아가면, OVERFLOW는 다시 가능하게 된다.

〔5〕 REVERT 문

REVERT 문은 하나 이상의 앞서 실행되는 ON 문의 효력을 말소시킨다. 이는 REVERT 문이 일어나는 블러크에 대해서 내적이며 그 블러크 동일 불러내기에서 실행되는 ON 문에만 영향을 끼친다. REVERT 문의 효과는 REVERT 문이 실행되는 동일 블러크에서 실행되는 호명된 조건의 어떤 ON 문의 효력도 말소시키는 것이다. 이는 그 블러크의 활동시에 효력이 있던 행위를 재설정 한다.

호명된 조건을 위해 행위가 지정되지 않은 블러크 안에서 실행되는 REVERT 문은 빈 문으로 취급된다.

보 기 :

```
(SIZE) : A : PROC ;
         DCL B ENTRY ;
         ON SIZE GO TO AERR ;
         . . .
         CALL B ;
         . . .
         END A ;
(SIZE) ; B : PROC ;
         ON SIZE GO TO BERR ;
```

```
. . .
ON SIZE GO TO CERR ;
. . .
REVERT SIZE;
. . .
RETURN;
END B;
```

이 보기에서, 만일 SIZE 오류가 B에서 첫째 ON문과 둘째 ON 문 사이에서 일어나면, 중단이 일어나고 통제는 BERR로 옮겨간다; 만일 SIZE 오류가 B의 둘째 ON과 REVERT사이에서 일어나면 중단이 일어나고 통제는 CERR로 간다; 만일 SIZE오류가 B에서 REVERT 문 다음에 일어나면, 중단이 일어나고 통제는 AERR로 간다. 그래서, REVERT 문은 B를 불러낸 점에 있는 (이는 A에서 CALL B 문이 실행되었던 점에 있던 것을 말함) SIZE를 위한 행위 지정을 재설정 한다.

〔 6 〕 SIGNAL 문

프로그래머는 SIGNAL 문으로써 ON 조건의 출현을 가장할 수 있다. 중단이 호명된 조건이 불능하게 되어 있지 않는 한 일어날 것이다. 이 문은 다음 형식을 가진다:

SIGNAL 조건-이름;

SIGNAL 문은 앞으로 올 이 지정된 조건을 위해 설정된 중단 행위를 실행케 한다. 이 문의 주요한 용도는 프로그램 검사에서 on-단위의 행위를 시험하고, 정확한 행위가 조건과 연관되게 결정하기 위한 것이다.

만약 신호된 조건이 가능하게 되어 있지 않으면, SIGNAL 문은 빈 문으로서 취급 된다.

제 12 장 기초된 변수와 지침변수
(基礎된 変数와 指針変数 based variable and pointer variable)

PL/I 프로그램에서 사용되는 각 표식어에 대해, 편성자는 정확한 명령을 만들기 위해서 이름과 연관된 속성을 결정할 수 있어야 한다.

보기로 :

A = B + C ;

만약 A, B, C가 부동점 변수이면, 부동점 명령이 편성되고; 만약 이들이 고정점수이면, 고정점 명령이 편성될 것이다.

이에 덧붙여, 편성자는 또 각 연산항의 주소를 결정할 수 있어야 한다. 어떤 경우에는, 편성자는 프로그램이 실행될 때 주소를 결정할 명령을 만들어야 한다. 변수의 기억소 종류는 주소가 취득되는 방법을 결정한다. 여기에는 세가지 독특한 경우가 있다.

① 정적 기억소 (静的 記憶所 static storage). 고정된 원점(原点 origin)으로 부터의 편차(offset 偏差)가 프로그램이 실릴 때 결정될 수 있다.

② 자동적 기억소(自動的 記憶所 automatic storage). 원점과 편차가 블러크에 들어갈 때 결정된다.

③ 기초된 기억소(基礎된 記憶所 based storage). 다른 기억소 종류와 어울려서, 요소가 인용될 때 사용되는 주소는 조직에 의해 결정된다. 사실 PL/I과 같은 언어를 사용하는 주된 잇점의 하나는 프로그래머가 주소와 주소 계산에 직접 관여할 필요가 없다는 것이다. 그러나, PL/I의 포

괄적 의도를 유지하기 위해, 설비는 프로그래머에게 주소를 통제할 수 있게 한다.

이 장은 세번째 종류(種類 class), 기초된 기억소와 주소 다루기를 취급한다.

제1절 지침 변수(指針 変数 point variable)

변수의 특이한 형식인 지침변수(pointer variable)는 PL/I에서 주소를 지정하는데 사용된다. 어떤 실부에서는 지침 변수가 실제로 주소를 포함하지 않으나, 이는 기억소에 자료를 갖다 놓을 때 사용된다; 사실 따지고 보면, 이는 하나의 주소로 볼 수 있다.

제2절 기초된 변수(基礎된 変数 based variable)

기초된 변수(基礎된 変数 based variable)는 연관된 지침 변수의 값에 따라서 기억소 내의 다른 자리에 적용될 수 있는 자료의 기술이다. 기초된 변수를 사용해서, 프로그래머는 (1)변수에 접근할 때 사용되는 주소를 명시로(明示로) 지정하고, (2) RECOSD-지향 입력/출력에 의해 전송되게 변수의 기억역을 위치시킬 수 있다.

기초된 변수가 선언될 때는, 이는 언제나 명시 선언된 지침 변수(指針 変数 pointer variable)와 연관되어야 한다.

선언의 형식은:

표식어 BASED (지침-변수)

보 기 :

```
DCL  P  POINTER ;    DCL  A  BASED ( P ) ;
```

A를 인용할 때마다, 주소는 지침 변수의 값을 사용해서 유도해 내야 한다. 사용된 지침변수는 기초된 변수의 선언에서 나타난 것이다. 이 보기에서는 p이다.

보기로 :

```
A = A + 2 ;
```

이 문에서, A의 주소를 결정하기에 사용된 지침은 양편 모두 P이다.

연관된 지침 변수가 유효한 값이 되는 동안은 기초된 변수에 관한 어떤 인용도 지침 변수에 의해 지적된 자리에 할당된 것처럼 취급된다.

1. 지침 지정 (指針 指定 pointer specificution)

지침 변수는 기초된 변수를 명명한 DECLARE 문에서 기초된 변수와 연관되어야 한다. 지정된 지침 변수는 POINTER 속성을 가지고 어디선가 명시로 선언된 것이라야 한다.

D.-편성자에 의해 부과된 제한은 기초된 변수의 선언에 사용된 지침 이름은 첨자가 없고, 수식 안된 요소 변수이어야 한다.

지침의 배열은 허용되고, 지침은 구조체의 요소가 될 수 있다. 그러나 이들 지침은 선언에서 기초된 변수와 연관될 수 없다.

2. 지침 변수의 값

기초된 변수가 인용되기에 앞서, 이것과 연관될 지침 변수에 값이 주어져야 한다. 이는 네가지 방법으로 될 수 있다 :

READ 나 LOCATE 문의 SET 선택항으로; 다른 지침 값의 대입으로; ADDR 내조립 함수에 의한 값의 대입으로.

〔1〕 READ 와 SET

SET 선택항을 가진 READ 문은 기록마디를 완충역으로 읽어드리고 지정된 지침 변수를 완충역을 가르키게 한다. 그 지침을 가지고 선언된 기초된 변수는 기록마디의 난을 인용하는 데 사용될 수 있다. 완충역에 있는 기록마디를 기술하기 위해 사용되는 기초된 변수는 완충역에 중첩되는 효과를 가진다. 기초 변수의 요소에 관한 인용 결과는 마치 기록마디가 기술된 구조체에 직접 읽혀들인 것과 같다.

〔2〕 LOCATE 와 SET

언제나 SET 선택항을 가져야 하는 LOCATE 문은 출력 완충역에서 기초된 변수에 기억소를 할당한다. 이 작용은 READ 와 SET 의 그것과 비슷하다. 거기서 기초된 변수는 완충역에 중첩된다. 그러나 이 경우에는, 기초된 변수의 요소의 기술에 관계되는 위치에 있는 완충역으로 자료를 이동시키기에 사용된다.

〔3〕 지침 값의 대입

지침 변수의 값은 단순 대입 문에서 다른 지침 변수에 대입될 수 있다. Q 와 P가 지침 변수이며 P가 유효한 지침 값을 가진다고 하자.

Q = P ;

이 문은 Q가 P을 가르키는 동일 자리를 가르키도록 세워진다. 연관된 지침으로서 P 또는 Q 어느 것을 사용한 지침 변수에 대한 인용도 기억소의 동일 자리를 인용하는 것이 된다. 또 지침 변수는 지침 값을 돌려보내는 프로그래머가 정의한 함수를 인

용함으로써 지침 값에 대입될 수 있다. 그러므로, 위 보기에서, P는 지침 값을 돌려보내는 프로그래머가 정의한 함수가 될 수도 있다.

〔4〕 ADDR 함수 값의 대입

ADDR 내조립 함수로 돌아오는 값은 함수인용의 인수로 명명된 변수의 위치를 지정하는 유효한 지침 변수 값이다.

보기로 :

```
P = ADDR(A);
```

이 문의 실행은 이것이 자료 변수 A의 위치를 가르키도록 지침 변수 P에 값을 줄 것이다. ADDR 함수 인용의 값은 지침 변수에만 대입될 수 있다. ADDR 함수 인용의 인수는 요소, 배열의 요소, 대구조체, 소구조체, 구조체의 요소가 될 수 있다.

ADDR 함수는 기초된 변수가 아닌 것을 가르키도록 지침을 세우기 위하여 사용될 수 있으므로, 이 설비는 기초된 변수가 아닌 것의 값을 인용하기 위해 기초된 변수를 사용할 수 있게 한다.

3. 지침 변수의 선언

지침 변수는 DECLARE 문에서 POINTER 속성을 가지고 명시 선언되어야 한다. 지침의 배열이 선언될 수 있다. 또 구조체의 요소 이름이 지침 변수로 선언될 수 있다. 태만에 의해, 지침 변수는 AUTOMATIC 기억소 종류 속성을 받는다. 그러나 STATIC 가 선언되어도 된다. 지침 변수는 BASED 속성을 갖지 못한다.

보 기 :

```
DCL  A  POINTER,
1  ELEMENT, 2  P  POINTER, 2  C  CHAR(10),  X(10)
```

```
POINTER  STATIC ;
```

주 : 지침 배열 변수는 이것이 SET 선택항에서 사용될 때는 단일 요소를 가르키도록 첨자가 붙어야 한다.

4 . 지침 변수의 제한

지침은 주소와 매우 밀접한 관계가 있으므로, 이것의 값은 이것이 사용되는 실황에 상당히 좌우된다. 현실 의존을 줄이기 위해서는, 어떤 제한이 지침 변수의 사용에 가해진다.

① 지침 변수는 연산자 =과 ¬=에 의해 지정된 비교 연산을 빼고는 어떤 연산의 연산항도 되지 못한다.

② 지침 변수 값의 대입은 다른 지침 변수에 대해서만 행해진다.

③ 지침 변수는 STREAM 입력과 출력에는 사용되지 못한다. RECORD 입력과 출력에서 사용될 때, 출력으로 쓰여진 지침 값은 만일 이것이 읽혀들여지는 때면 동일 자료를 위치시키도록 되지 못한다.

제3절 기초된 기억소와 지침의 이용

기초된 기억소와 지침 취급 설비는 일차로 입력과 출력 완충역에서 기록다디를 처리할 수 있게 하는 것이다. 이는 현저히 기억소를 절약할 수 있게 한다. 특히 한 화일에 여러 종류의 기록다디가 있을 때 그렇다.

여러가지 상이한 기초 변수의 선언이 하나의 지침에 연관될 수 있다.

보 기 :

```
DCL     P  POINTER, 1  ISSUE  BASED (P),
        2  CODE  CHAR(1),2  PARTNO  PIC'(7)9',
        2  QTy   PIC'(4)9',2  DEPT  PIC'99',
        2  JOBNO  PIC'(4)9',
        1  RECEIPT  BASED(P),2  CODE  CHAR(1),
        2  PARTNO  PIC'(7)9',2  QTy  PIC'(4)9',
        2  SUPPLIER  PIC'(5)';
READ    FILE (TRANS)  SET(P);
IF      ISSUE·CODE = 'R' THEN  GO  TO  RL1;
IF      SUPPLIER > 1000  THEN  GO  TO  INHS1;
```

이 보기에서, 두개 기록마디 기술 ISSUE와 RECEIPT는 동일 지침과 연관되어 있다. 한번 P가 SET 선택항을 가진 READ 문의 실행에 의해 값을 받으면, 두 기록마디 중의 어느 것이라도 인용이 가능하다. 이 지침은 완충역 안의 자리를 인용한다.

기록마디는 또 문자 줄과 수치 문자난 이외의 변수로 구성될 수 있다. 몇개의 기록마디라도 한개 지침과 연관될 수 있다. 지침이 값을 받으면, 모든 기록마디는 동일 기억소를 인용하고 중첩된다. 그런 기록마디의 기술의 중첩은 기계와의 독립성이 되고 조심스러이 사용되어야 한다.

1. 가변 길이 매개변수 나열 (可変 길이 媒介変数 羅列 variable-length parameter lists)

PL/I에서, 프로그래머가 짠 수속은 정해진 수의 매개변수만 가질 수 있다. 또 매개변수는 지정되어야 한다. 인수는 주소를

보내므로써 인수와 연관된다 (가인수의 주소일 때도 있다) . 지침 배열을 단일 인수로 보내므로써 , 이는 가변 길이 매개변수 나열로 할 수 있다. 그러면 배열 요소의 일부가 사용안 될 있다.

다음 수속은 값이 두 경계 사이에 드는 가를 검사한다. 수속은 두 매개변수를 가진다. 검사될 값과 두 지침의 배열 . 첫째 지침은 상한 , 둘째는 하한을 지정한다. 두 경계에 들지 않으면 , 연관된 지침은 돌려보내질 때 무효가 된다. ("지침 취급" 의 NULL 을 보라) .

```
LIMIT : PROC(X,P) BIT(1);
  DCL P(2) POINTER,(P1,P2) POINTER,
  TOP BASED(P1), BOTTOM BASED(P2);
IF P(1) ¬=NULL THEN DO; P1=P(1);
IF X>=TOP THEN RETURN('0'B); END;
IF P(2) ¬=NULL THEN DO; D2=P(2);
IF X<=BOTTOM THEN RETURN('0'B); END;
RETURN ('1'B);
END LIMIT;
```

LIMIT를 부른 수속이 다음과 같다고 하자 :

```
DCL LIMIT RETURNS BIT(1), Q(2), POINTER;
Q(1)=ADDR(HIGH);
Q(2)=NULL;
IF LIMIT(y,Q) THEN DO;
```

기초된 변수 선언에 있는 지침은 첨자가 붙지 못하므로 , 경계 TOP 와 BOTTOM 을 인용하기에 사용되는 두개의 다른 지침 변수를 정의하는 것이 필요함을 유의하라 .

수속 LIMIT가 정도 15의 고정점 이진수를 돌려보내지 않으므로(첫자의 태만에 의해), 이는 불러내는 수속에서 RETURUS 속성으로 선언되었다. 불러내는 수속에서, 값이 내조립 함수 ADDR와 NULL을 사용해서 Q(1)과 Q(2)에 대입되었다. 수속은 IF 문에 있는 함수 인용에 의해 불러내진다.

제4절 지침 취급(指針 取扱 Pointer manipulation)

두개 중요한 내조립 함수가 지침 변수를 취급하기에 사용될 수 있게 준비되어 있다. 이들은 ADDR 내조립 함수와 NULL 내조립 함수이다.

ADDR 내조립 함수는 이미 간단히 논술되었다. 이는 변수 이름인 하나의 인수를 요하며, 변수를 가르키는 값을 돌려보낸다. 이는 요소 변수, 배열 변수, 배열의 요소, 대구조체, 소구조체, 구조체의 주소를 발견하는데 사용된다.

ADDR 함수는 기초 변수 또는 기초 변수 아닌 인수의 주소를 가르키는 값을 돌려보낸다. 배열이나 구조체 인수일 때는, 이들의 첫번째 요소의 ADDR와 같음을 유의하라.

보 기:

```
DCL  P  POINTER ;  B(10,10)  BASED(P),  A(10,10);
```

ADDR(A(1,1))은 ADDR(A)와 같고, 다음의 문

```
P = ADDR(A);
```

에서, B(1,1)은 A의 첫번째 요소를 인용하는 것이 될 것이다.

두번째는 NULL 함수이다. 이는 함수 인용에서 인수를 요하지 않는다. 이는 무효한 지침 값을 보낸다. 이것은 유효한 주소를 인용하지 못하는 값을 말한다.

STANDARD Coding Form

PROGRAM | PUNCHING INSTRUCTIONS: GRAPHIC / PUNCH | PAGE / OF | CARD ELECTRO NUMBER

PROGRAMMER 1. PL/1 프 로 그 램 | DATE

표 13-1. PL/1 프로그램 STATEMENT | Identification Sequence

```
C72A3:   PROCEDURE OPTIONS (MAIN);                                          ∅1
         DECLARE PAGE_NO FIXED DECIMAL,                                     ∅2
                 1 CARDIN                                                   ∅3
                   2 ACCNT NO picture '(8)9',                               ∅4
                   2 NAME CHARACTER (25),                                   ∅5
                   2 ADDRESS CHARACTER (25),                                ∅6
                   2 PAYMENT picture '$$$$9V.99',                           ∅7
                   2 REST CHARACTER (14),                                   ∅8
                 1 B,                                                       ∅9
                   2 BALANCE picture '$$$$$9V.9R',                          1∅
                   2 REST1 CHARACTER (71),                                  11
                 WRKA CHARACTER (8) DEFINED CARDIN,                         12
                 WRKB CHARACTER (9) DEFINED B,                              13
                 PAYMNT FILE INPUT RECORD ENVIRONMENT                       14
                          (CONSECUTIVE F(8∅) MEDium(SYS∅∅3, 254∅)),         15
                 ACCNTS FILE UPDATE RECORD DIRECT KEYED                     16
                          ENVIRONMENT (REGIONAL(1) F(8∅)                    17
                                       MEDium(SYS∅∅1, 2311)),               18
                 EXCP FILE STREAM OUTPUT PRINT                              19
                          ENVIROUMENT (MEDium(SYS∅∅2, 14∅3) F(133));        2∅
         OPEN FILE (PAYMNT), FILE (ACCNTS);                                 21
         ON ENDFILE (PAYMNT) GO TO EOF;                                     22
/* SET UP PAGE CONTROL */                                                   23
         ON ENDPAGE (EXCP) GO TO NEW_PAGE;                                  24
```

STANDARD Coding Form

PROGRAM | PUNCHING INSTRUCTIONS: GRAPHIC / PUNCH | PAGE 2 OF | CARD ELECTRO NUMBER

PROGRAMMER | DATE

STATEMENT | Identification Sequence

```
/* SET UP HEADING AND THEN PRINT HEADINGS FOR FIRST PAGE */                    25
        PAGE-NO = 0;                                                           26
NEW-PAGE: PAGE-NO = PAGE-NO + 1;                                               27
        PUT FILE (EXCP) PAGE LINE(3) EDIT ('PAGE ', PAGE-NO)                   28
                                          (X(10), A(5), F(5));                 29
        PUT FILE (EXCP) LINE(5) EDIT ('NO PAYMENT RECEIVED FROM')              30
                                     (X(30), A(24));                           31
        PUT FILE (EXCP) LINE(10)                                               32
            EDIT ('ACCOUNT NO', 'NAME', 'ADDRESS', 'BALANCE DUE')              33
                 (COLUMN(5), A(11), COLUMN(16), A(5),                          34
                  COLUMN(41), A(7), COLUMN(66), A(11));                        35
/* TEST TO SEE IF NEW-PAGE ENTERED ON INTERRUPT */                             36
        IF PAGE-NO = 1 THEN GO TO NEWCARD;                                     37
                       ELSE GO TO ZZ;                                          38
/* MAIN UPDATE LOOP */                                                        39
NEWCARD: READ FILE (PAYMENT) INTO (CARDIN);                                    40
        READ FILE (ACCNTS) INTO (B) KEY(ACCNT-NO);                             41
        IF PAYMENT = 0 THEN                                                    42
                   IF BALANCE<= 0 THEN GO TO NEWCARD;                          43
                                  ELSE                                         44
                     ZZ: PUT FILE (EXCP) SKIP EDIT                             45
                                 (WRKA, NAME, ADDRESS, WRKB)                   46
                                 (COLUMN(5), A(8),                             47
                                  COLUMN(16), A(25),                           48
```

STANDARD Coding Form

PROGRAM | PUNCHING INSTRUCTIONS: GRAPHIC / PUNCH | PAGE 3 OF | CARD ELECTRO NUMBER

PROGRAMMER | DATE

STATEMENT | Identification Sequence

```
                                                    COLUMN(41), A(25),                  49
                                                    COLUMN(66), A(9);                   50
                                ELSE DO;                                                51
                                     BALANCE=BALANCE-PAYMENT;                           52
                                     REWRITE FILE(ACCNTS) FROM(B) KEY(ACCNT-NO);        53
                                     END;                                               54
                          GO TO NEWCARD;                                                55
                EOF: CLOSE FILE(PAYMENT), FILE(ACCNTS), FILE(EXCP);                     56
END C72A3;                                                                              57
```

2. 보기 프로그램의 설명

그림 13-1은 완전한 PL/I 프로그램의 보기이다. 그러나 PL/I이 어떻게 작성되는가를 설명하기 위한 것이고 어떤 문제를 풀려고 의도된 것은 아님을 유의하라.

이 프로그램은 카드를 읽어서 지불이 되었는가를 검사하는 것이다. 만일 지불된 것이 있으면 지불액이 계정의 잔고에서 빼진다. 잔고는 별개의 화일에 있다. 만일 지불이 없고 또 잔고가 있으면 이름, 주소, 계정 번호, 잔고가 인쇄된다.

D-편성자는 원시 프로그램의 모든 카드의 첫째 자리는 빈자이어야 하고 73번 부터 80번 자리는 무시되며 어떤 내용으로 있어도 된다. 이 보기에서 79와 80에 카드 번호가 있다. 이런 자리 제한이 지켜지면 문은 어느 자리에서도 시작되고 끝날 수 있다. 문은 한 카드에서 다음 카드로 이어질 수도 있다. 상수가 2개 카드에 이어지면 먼저 카드의 마지막은 72자리가 되고 다음 카드의 처음은 2째 자리가 된다.

카드 1에 있는 PROCEDURE 문은 수속을 지명한다. MAIN 선택항은 프로그램의 개시 수속(開始 手続 initial procedure)을 지정하는 실무상 정해진 선택항이다.

카드 2부터 20의 DECLARE 문은 수속에서 사용되는 표식어의 속성을 선언한다.

카드 2의 PAGE-NO은 FIXED DECIMAL 속성을 받으며 정도가 없으므로 태만 정도(5, 0)을 받는다.

카드 3부터 8까지의 구조체 선언은 입력 기록마디 CARDIN을 기술한다. 이 기록마디는 처음의 8자리에 계정 번호를(이 계정 번호는 또한 계정 화일에서 계정을 찾는데 쓰이는 열쇠가 됨을

유의하라) 갖는다.

카드 9, 10, 11은 계정 화일(ACCNTS)로 부터 읽히며 잔고가 있는 기록마디를 기술하는 구조체를 선언한다. 각 기록마디는 잔고가 1부터 9까지 자리에 있다. 각 기록마디는 계정 번호인 열쇠에 의해 증명된다. BALANCE난의 맨 끝 R은 이것이 음수일 때는 -를 나타내는 것의 중복천공되어야 함을 유의하라.

카드 12와 13에서 WRKA와 WRKB가 CORDIN과 B에 정의된 문자 줄로서 선언되었다. 이는 인쇄하기 위해 수치 문자 변수를 문자 줄 변수로 바꿀 필요를 없애준다.

첫번째로 실행 가능한 문은 카드 21에 있다. EXCP는 카드 28의 PUT 문에 의해 암시로 열린다.