Programming

프로그래밍

유 길 호
전국대학교도서관

1. Soft Ware의 概要

전자계산기가 오늘날 각광을 받고 있는 것은 단순히 계산을 1초간에 수천, 수만번이나 할 수 있다던가 또는 1분간에 결과를 수백, 수천행씩이나 인쇄할 수 있다는 Hard Ware에만 의존하고 있는 것은 아니다.

「프로그램」 記憶式電子計算機는 실제의 일을 수행하기 전에 어떤 순서에 따라 처리해 나갈 것인가를 미리 전자계산기에 가르쳐 주어야 한다. 이 一連의 처리순서가 곧 「프로그램」이라고 하며 우리가 이 「프로그램」만 가르켜 주면 그 뒤는 전자계산기가 이에 따라 자동적으로 모든 일을 수행하게 되는 것이다. 따라서 아무리 복잡한 數式의 계산이건, 대량의 「데이타」처리든 또는 언어의 번역이나 문헌의 초록을 만드는 등등의 일이라도 「프로그램」만 만들어 넣어 준다면 그뒤는 전자계산기가 자동적으로 또한 극히 짧은 시간내에 이것을 처리할 수 있게 된다.

전자계산기의 Hard Ware라는 것은 어디까지나 記憶이라던가 演算, 制御등을 할 수 있는 장치와 기능을 갖고 있을 뿐이지 이것을 활용하여 올바른 기능을 발휘하도록 하거나 또는 기능을 살리지 못하고 죽이느냐 하는 것은 오로지 사람이 전자계산기를 교육하고 훈련하는 내용 곧 전자계산기의 사용에 관한 기술에 달려 있다고 하겠다.

이와같이 전자계산기를 효과적으로 활용하기 위하여 요구되는 모든 Programming System을 Soft Ware라 부른다. Soft Ware가 시원치 못한 계산기는 아무리 Hard Ware의 성능이 좋더라도 효과적인 이용을 기대할 수 없는 쇠덩어리에 지나지 않는다. 이러한 의미에서 오늘날 전자계산기는 Hard Ware보다는 Soft Ware가 더 중요시되고 있으며 Soft Ware에 대한 연구, 개발에 한층 더 많은 노력이 필요하다.

2. 「프로그래밍」의 작성 절차

전자계산기를 활용하여 수식계산을 하거나 자료처리를 하려면 우선 그것을 어떤 절차에 따라 계산을 하여야 할지 계산순서와 절차를 명확히 하여야 한다.

이를 위하여 첫단계로 풀려고 하는 문제를 인식하는 일이다. 풀려고하는 문제가 자료처리의 문제일 경우에는 이에 필요한 자료(Data)가 어떻게 되어 있는지 내용을 정확히 파악하고 문제에 포함된 여러 現象이나 행동을 數式으로 표현하거나 수학적인 모델(Model)을 만들어야 한다.

둘째로는 인식된 문제를 Flowchart로 표시하여야 한다. Flowchart란 "자료를 읽어 들인다" "계산한다" "비교한다" "결과를 인쇄한다" 등과 같이 계산의 절차를 순서대로 그림으로 표시한 것을 말한다.

Flowchart가 완성되면 세째단계로 전자계산기가 알아들을 수 있는 전자계산기언어(Computer Language)로 이를 번역하여야 한다. 이것을 符號化(Coding)라 한다. 우리가 외국사람과 대화를 할려면 그 나라 말을 할줄 알아야 하듯이 전자계산기를 활용하여 어떤 계산이나 자료처리를 할려면 전자계산기가 이해할 수 있는 언어를 알아야 한다.

언어에 대해선 추후 설명하기로 한다.

풀려고 하는 문제를 Flowchart에 따라 계산기언어로 Coding한 것을 Program이라고 하며 Program의 작성방법을 익혀 이것을 작성하는 것을 Programming이라 한다. Programming을 전문적으로 하는 사람을 Programmer라 부른다.

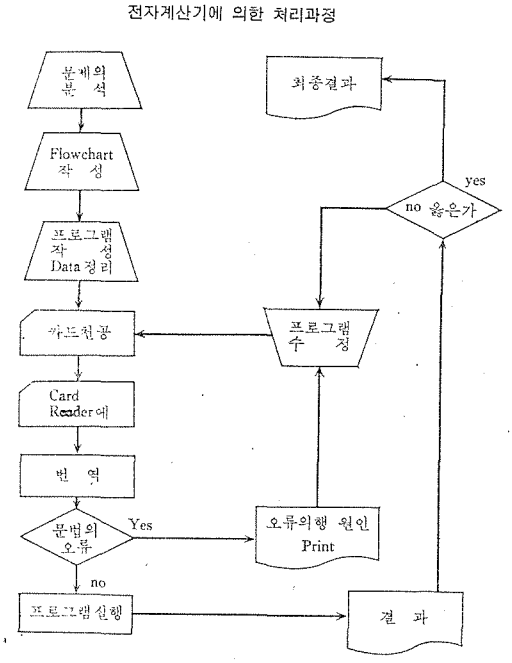
Coding이 끝나서 Program이 완성되면 이를 전자계산기에 입력시켜서 풀고자 하는 해답을 얻어야 한다. 즉 Program을 실행하여야 한다. 따라서 이것이 네째단계로 해야 할 일이다.

Program을 계산기에 넣어서 결과를 얻어내는 일은 전자계산소에 종사하는 사람(Key puncher, Operator, Clerk)이 하는 일이고 전자계산기를 이용하는 사람으로서 이 단계에서 해야 할 일은 다음과 같다. 즉 만일 해답이 제대로 안나온 경우 Program에 틀린곳이 있는가 없는가 하는 것을 검토하여 수정하는 일이다. 이것

을 Debugging한다고 한다

해답이 제대로 안나오는 것은 Coding과정에서 틀리는 경우도 있지만 문제를 잘못 인식하여 수학모델을 잘못 定立하든가 Flowchart를 잘못 작성하였기 때문에 오는 경우도 있으므로 Debugging할 때에는 제1단계부터 3단계까지의 일을 반복하여 실시하여야 한다.

Program에 과오가 있을 경우에 Program의 출력(Output)에 어떠한 과오 때문에 Program의 실행이 안된다는 내용의 Message가 찍혀 나오게 되어 있다. 이것을 Error Message라 한다.



3. Flowchart의 작성

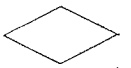
Flowchart는 문제를 분석하여 전자계산기가 능률적으로 시행할 수 있도록 그 처리과정이나 처리순서를 정하고 이들 상호간의 관계를 일정한 기초를 사용하여 일목요연하게 표시한 그림을 말하며, 이에는 System Flowchart와 Program Flowchart가 있다. Flowchart만 정확히 작성되면 Program의 Coding은 Flowchart에 따라서 계산기언어로 바꾸어 적으면 충분하게 될 정도이다. 그 뿐아니라 Flowchart는 Program의 정확성 여부를 검토하는 데에 기본적인 자료가 되며, Test Run의 결과 Program에 이상이 있으면 이를 수정하기가 용이하다.

System Flowchart는 처리할 문제의 여러 과업간의 상호관계를 표시해 주는 도표이다. 즉 처리할 문제에 있어서 계산기 처리부분과 수동식 처리부분을 구분하고 이들 상호관계를 나타내 주는 도표이다.

Program Flowchart는 계산기가 처리하기에 알맞게 상세히 처리과정과 순서를 표시한 그림이다. 즉 전자계산기 내부에서 처리되는 과정을 그대로 표시하는 그림이라 할 수 있다.

Program Flowchart에 사용되는 부호는 아래와 같다

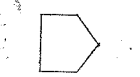

여러가지 처리기능(Processing)

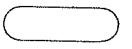

비교, 판단, 결정을 표시(Decision)


Flowchart에 자세히 설정하지 않고, 외부에서 미리 결정된 사항의 표시(Predefined process)


Flowchart간의 연결(Connector)


입력 및 출력(Input/Output)

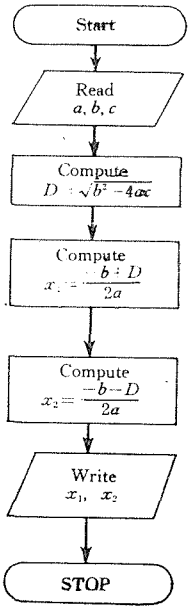

프로그램의 수정(Program modification)


개시, 종료의 표시(Terminal)


보고서 및 결과(Document)

기호와 기호의 연결(Flow)

例) 2차방정식 $3x^2+8x+5=0$의 근을 계산하는 Program을 작성하고자 할때의 Flowchart.



4. 「프로그래밍」의 실행과정

계산하고자 하는 문제에 대한 Flowchart가 작성되고 Coding이 완료되면 즉 Program이 완성되면 이것을 전자계산기에 넣어 Program을 실행하여야 한다. 그런데 전자계산기가 알아들을 수 있는 언어는 機械語뿐이기 때문에 機械語로 부호화가 된 Program인 경우에는 바로 실행이 되지만 그 이외의 언어로 부호화가 된 Program인 경우에는 이것을 전자계산기가 알아들을 수 있는 언어로 번역을 해야 된다.

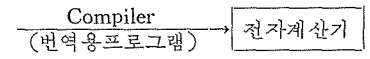
따라서 Assembly로 부호화된 Program은 Assembler에 의해서, 문제중심언어(Compiler Language)로 부호화된 Program은 Compiler에 의해서 각각 기계어로 번역이 된다.

물론 여기서 Assembler와 Compiler는 어떠한 기계장치가 아니고 기계어로 번역하는 하나의 Program이다. 그러므로 전자계산기는 실제로 2가지의 일을 하고 있는 것이라고 보아야 한다.

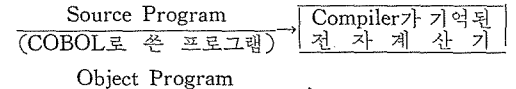
첫째는 Assembly언어나 Compiler언어로 쓴 Program을 전자계산기에 이미 기억되어 있는 Assembler나 Compiler에 의하여 기계어로 번역하는 일이고, 둘째는 이렇게 번역한 프로그램을 실행하는 일이다. 따라서 Assembly언어나 Compiler언어로 된 Program을 Source Program이라 하고 이것이 전자계산기에 의하여 기계어로 번역된 Program을 Object Program이라 부른다. 또한 Compiler와 Assembler를 총칭하여 Processor라고도 부른다. Processor라고 하면 전자계산기 자체를 말할 때도 있고 Compiler와 Assembler를 총칭하여 Source Program을 Object Program으로 번역하는 Program을 말할 때도 있다.

그림 : 프로그래밍의 실행과정

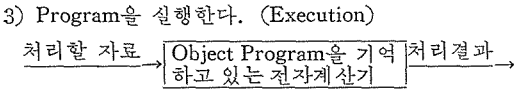
1) Compiler(또는 Assembler)을 기억시킨다. (Compiling)



2) Source Program을 Object Program으로 번역 (Translation)



3) Program을 실행한다. (Execution)



5. 「프로그래밍」의 언어

전자계산기에 의하여 자료처리나 수식계산을 할려면 전자계산기가 알아들을 수 있는 언어를 사용해야 한다. 풀려고 하는 문제에 대한 Flowchart를 전자계산기가 알아들을 수 있는 언어로 바꾸어 놓아야 한다. 이것을 Coding이라 하는데 Coding을 하려면 자기가 사용할려고 하는 전자계산기가 어떤 종류의 언어를 사용하는지를 먼저 파악해야 한다. 이와같이 전자계산기가 알아들을 수 있는 언어를 Programming Language라 부른다.

1) 機械語(Machine Language)

원칙적으로 전자계산기가 알아들을 수 있는 언어는 한가지 밖에 없다.

그것을 기계어라고 하는데 이것은 전자계산기를 만들 때 이미 그 전자계산기를 사용하는 사람들을 위해 정해져 나오는 숫자로 된 符語로서, 언어자체가 완전히 기계중심의 숫자부호(Machine Code)로 만들어져 있기 때문에 배우기가 어려울뿐 아니라 기계내용을 잘 알아야 하므로 부호화하는데 즉 Program을 작성하는데 시간이 많이 걸린다. 그러나 기계어는 일단 Program이 완성되어 전자계산기에 입력되면 Program의 실행시간(Processing Time)이 가장 짧은 것이 특징이다.

2) Assembly언어

기계어의 상징부호(Symbolic Code)화한 것이 Assembly 언어이다. 이것 또한 기계중심으로 만든 언어이기 때문에 사용하는데는 기계어 보다는 쉬우나 기계어와 마찬가지로 불편한 점이 많다.

전자계산기는 기계어 밖에는 이해하지 못하므로 Assemly언어에 의하여 작성된 Program은 다시 기계어로 번역해야 한다. 전자계산기는 기종마다 구조가 다르므로 기계마다 Assembly언어가 다르다. 또한 Assembly언어도 숫자를 기호화한 것이므로 이를 배우는 데에는 어느정도 기계(Hard Ware)에 대한 지식이 필요하다.

3) 문제중심언어(Compiler Language)

Assembly언어는 기계어에 비하면 배우기가 쉽지만 아직도 기호를 사용한다는 단점이 있다. 이 기호를 배재하여 일상언어를 사용하고 계산기에 대한 지식이 없어도 쉽사리 Pogram을 작성할 수 있는 언어가 요청되어 개발된 것이 Compiler Language이다. Compiler언어는 일상언어로 만들어지는 「프로그래밍」언어로서 기종에 상관없이 어느 기종에나 사용할 수 있다.

Compiler언어도 작성된 Program은 Compiler라는 번역프로그램을 통하여 기계어도 번역, 계산기로 하여금 처리케 한다. Compiler언어에는 다음과 같은 4가지 언어가 가장 많이 사용되고 있다.

가. COBOL(Common Business Oriented Language)

Common은 "공통적" 또는 "일반적"이라고 번역될 수 있는 말로서 COBOL이 공통적이고도 일반적인 언

어임을 뜻한다. "공통적"이란 기종에 구애됨이 없이 모든 전자계산기에 사용가능하다는 뜻이고 "일반적"이란 보통언어 즉 특수언어나 힘든 언어가 아닌 평범한 언어를 뜻하는 것이다. 또한 Business Oriented란 사무용으로 되어 있다는 뜻으로 과학용과 대립되는 개념이다. 사무용은 "多量의 入出力과 단순한 처리조작"이고 과학용은 "小量의 入出力과 복잡한 처리조작"이다. 따라서 사무용이란 처리조작이 비교적 단순한 업무에 알맞도록 만든 언어이다.

이 언어의 특징은 영어와 유사하다는 점에 있으며 영어를 아는 사람은 쉽게 읽힐 수 있다는 점이다. COBOL에 대한 설명은 다음장에서 설명하기로 한다.

나. FORTRAN(FORmula TRANslator)

FORTRAN은 수식계산이나 공학(또는 설계) 문제를 풀기에 적합하도록 만든 언어이다. 따라서 이 언어는 과학기술계산용으로 많이 사용되며 간혹 사무용에도 사용된다. FORTRAN System도 COBOL과 같이 Compiler System의 일종이므로 문법상의 차이만을 제외하고는 모든 특성이 COBOL과 동일하다.

FORTRAN도 다른 언어와 마찬가지로 사용할 수 있는 정해진 문자기호가 있다.

문자기호(Character Set)에는 ① 문자(Letters) : A~Z까지의 26개 ② 숫자(Numbers) : 0~9까지의 10개 ③ 특수기호(Special Character) : +(더하기), −(빼기), ✻(Asterisk), /(Slash), .(종지부), ,(구점점), ( )(왼쪽 및 오른쪽 괄호), =(등호), '(Apostrophe), $(달라기호) 총 11개 이상의 47개 문자기호만을 사용한다.

숫자에 있어서는 소형숫자를 사용하지 않기 때문에 우리가 흔히 쓰는 x_1, x_2와 같은 添字(Subscript)을 사용할 수 없다. 따라서 x_1은 $x(1)$으로, x_2는 $x(2)$와 같이 괄호를 열고 添字를 표시한다. 수학에서 많이 사용하는 〈, 〉, ≤, ≥, ∪(Union), ∩(interset), 중괄호, 대괄호 같은 특수기호가 없다.

따라서 〈은 .LT.로 〉은 .GT., ≤은 .LE., ≥은 .GE., ∪은 .AND., ∩은 .OR.로 표시한다.

중괄호와 대괄호는 소괄호를 그대로 사용한다. FORTRAN에서 =(등호)는 수학에서 뜻하는 등호가 아니라 우변의 값을 좌변의 값으로 "대치한다"의 뜻으로 사용된다.

ax^2+bx+c ——→A✻X✻✻2+B✻X+C

$a[x+b(x+c)]$ ——→A✻(X+B✻(X+C))

※ Program例

A상자에 들어있는 120개의 사과와 B상자에 들어있는 210개의 사과를 합쳐서 C상자에 담고, C상자의 사과의 합을 계산하는 문제

```
      READ (13, 1) A,B
    1 FORMAT (F3.0, F3.0)
      C=A+B
      WRITE (10, 2) C
    2 FORMAT (F3.0)
      STOP
      END
```

다. ALGOL(ALGOrithmic Langnage)

FORTRAN이 과학기술계산용으로 미국에서 널리 사용되고 있었는데 반하여 이것은 유럽에서 과학기술계산용으로 개발되어 널리 사용되고 있는 과학기술계산용 언어이다. 따라서 용도상으로는 FORTRAN과 차이가 없으나 ALGOL에서 사용하는 수학기호는 FORTRAN에 없는 것, 즉 〈, 〉, ≤, U(Union 혹은 AND), ∩(Intersect 혹은 OR) 등이 있다. 그러나 FORTRAN에서는 이러한 수학기호를 직접 사용하지는 않으나 그의 표현은 가능하므로 차이점이라 할수는 없다.

라. PL/1(Programming Language-1)

PL/1은 비교적 최근에 개발된 언어로서 사무처리용인 COBOL과 과학기술계산용인 FORTRAN의 특징을 병합하여 사무처리와 과학기술계산에 다같이 사용할 수 있도록 만든 언어이며 1이라는 번호를 붙인 것은 언어가 개정될 때에 대비한 것이다.

6. COBOL 프로그래밍

COBOL에서 사용되는 글자는 다음과 같다.

① 숫자 : 0~9(10자)

② 영문자의 대문자 : A~Z(26자)

③ 특수글자 : 15자

특수글자	명 칭	특수글자	명 칭
+	Plus	−	Minus 혹은 Hyphen
✻	Asterisk	/	Slash
=	Equal	>	Greater than
<	less than	,	Comma
.	Period	;	Semicolon
"	Quotation mark	(	Left Parenthesis
$	Dollar	)	Right 〃
	Blank 혹은 Space		

COBOL의 글자중 숫자는 Numeric이라 하고 영문자는 Alphabetic이라 한다. 숫자, 영문자, 특수글자를 모두 합쳐서 Alphanumeric 혹은 Alphameric이라 한다.

COBOL프로그램에 있어서 의미와 용법이 정해져 있는 단어를 Reserved Word라 하여 이 단어는 정해진 의미로 정해진 용도에만 사용해야지 그렇지 않으면 계산기에는 Error가 발생되어 제대로 처리하지 못하거나 전혀 실행을 하지 않는다.

COBOL프로그램은 COBOL Reserved Word를 사용

하여 작성하는데 이것만으로는 완전한 프로그램을 작성할 수가 없다. User-defined word는 프로그래머가 만든 단어라는 뜻으로서, 프로그램을 작성함에 있어서 프로그래머가 임의로 만든 단어이다.

User-defined word는 일정한 규칙에 따라 만드는 바, 작성법칙은 다음과 같다.

① 숫자와 영문자를 사용한다.

② 특수글자는 사용할 수 없다. (단, —(Hyphen)은 사용)

③ 하나의 단어가 30자를 초과할 수 없다.

④ Hyphen은 단어의 맨처음과 맨끝에는 올 수 없다.

⑤ 숫자와 Hyphen만으로는 단어를 만들 수 없으며 적어도 하나만은 영문자가 있어야 한다.

⑥ 하나의 단어를 두가지 또는 그 이상의 의미로 사용해선 안된다.

⑦ 단어사이에 공백이나 빈칸이 있으면 2개의 단어로 간주함으로 하나의 단어는 언제나 붙여 써야 한다.

1) 연산문과 이송문

가. ADD문

```
ADD  3   4   GIVING A.
ADD  X   Y   GIVING Z.
ADD  15  26  GIVING HAPGYE
ADD BONGGUP SUDANG GIVING WOLGUP.
```

나. SUBTRACT문

```
SUBTRACT 5 FROM 8 GIVING F.
SUBTRACT SEGUM FROM CHONGAIK GIVING W.
```

다. MULTIPLY문

```
MULTIPLY 3 BY 4 GIVING T.
MULTIPLY SIGAN BY IMMYUL GIVING BONGGUP.
```

라. DIVIDE문

```
DIVIDE 3 INTO 9 GIVING K.
DIVIDE 5 INTO WONGA GIVING BIYONG.
```

마. MOVE문

```
MOVE 56 TO SIGAN.
MOVE BONG-GUP TO WOLGUP.
MOVE 4 TO A B C.
```

※

```
MOVE 40 TO SIGAN.
MOVE 2.5 TO IMMYUL.
MULTIPLY SIGAN BY IMMYUL GIVING GROSS.
MULTIPLY GROSS BY 0.20 GIVING SEGUM.
SUBTRACT SEGUM FROM GROSS GIVING WOLGUP.
```

2) 입력 및 출력

```
ACCEPT JAGEOP-SIGAN.
ACCEPT IRUM.
DISPLAY JIGUP.
```

※ 예제 프로그램

상품의 원가를 계산하라. 원가는 구입가격뿐 아니라 운송비도 고려함.

문제분석 : ① 구입비=개당 구입가격×수량
운송비=개당 운송비×수량
원 가=구입비+운송비

② Data 이름

개당구입가격 : PRICE
수 량 : SURYANG
개당운송비 : UNSONGBI
원 가 : COST라 하고 연산과정에서 임시로 필요한 Data이름인 구입비와 운송비는 각각 P와 U라 한다.

③ 개당구입가격, 개당운송비, 수량 및 원가를 출력시킨다.

「韓國十進分類法」 개정에 따른 意見接受

本會는 "韓國十進分類法(KDC)" 개정작업을 추진하여 今年中 그 개정 3版을 내놓을 豫定이었으나 보다 充實한 개정판을 내놓고자 이에 각급도서관 分類擔當者들의 意見을 綜合하여 개정판에 적극 반영코자 하오니 깊은 관심을 가지고 "KDC"를 再檢討하여 다음 內容에 대한 意見을 79年 1月 30日까지 제출하여 주시기 바랍니다.

1. 새로운 主題 및 필요한 項目이 빠진 경우
   (1) 그 內容과 (2) 이를 어느 項目에 넣으면 適合한가
2. 각 項目中 展開를 보다 상세히 할 部分과 그 內容
3. 發見된 誤字와 특히 索引中의 잘못된 分類番號
4. 例가 잘못된 부분(특히 분류번호 등)
5. 기타 「KDC」사용中 不充分한 부분이 발견된 점