韓國數學教育學會誌<數學教育>
1983. 6. Vol. XXI, No. 3

Journal of the Korea Society of Mathematical Education
Jun. 1983. Vol. XXI, No. 3

高校數學教育課程의 順序圖의 指導에 關한 硏究

金 炳 喆·金 命 烈

1. 序 論

逆說的이지만 컴퓨터에 關한 最小限의 知識을 가지고 最大限의 敎授 效果를 爲하여 이미 컴퓨터를 알고 있는 指導敎師가 看過하기 쉬운 점을 指摘하여 初心者인 學生들의 理解의 障碍物을 除去하고자 하며 初中高校의 電算敎育의 前哨的 役割로 冷徹한 反省과 새로운 契機를 마련코자 한다. 우선 高校數學 敎科書에 있는 順序圖(흐름도 : Flowchart)의 보다 올바르고 쉬운 指導를 위한 몇가지 具體的 方案을 提示한다.

2. 順序圖의 意義

주어진 問題를 解決하는 處理過程에 어떤 規則을 發見한 그 解決方法을 Algorithm이라고 하며 이것을 表現하는 데는 ① Pseudocodes ② Flowcharts ③ Schematic logic의 3가지 基本的 技法이 있다.[1] 그 외에도 여러 가지 方法들이 硏究되어 왔다.[2] 이들 方法은 모두 프로그램의 開發道具로 利用되고 있지만 그 優越成은 個人의 選擇에 依存한다.

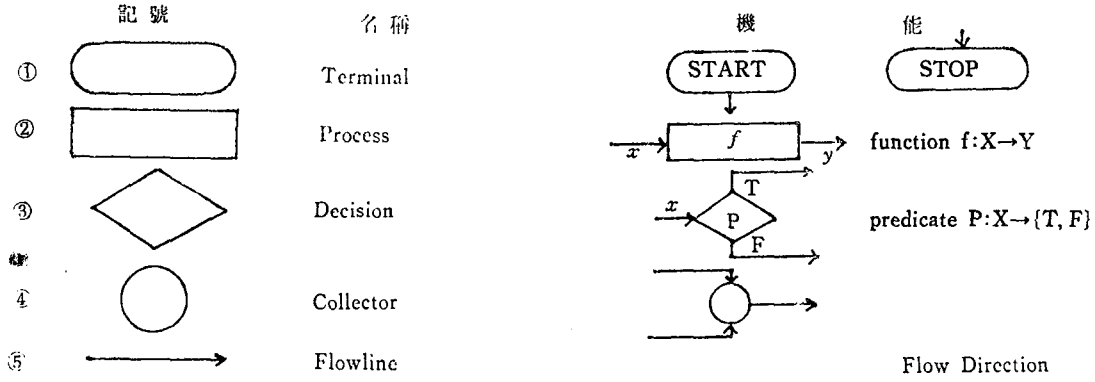
Algorithm을 特定 프로그래밍 言語로 變換한 것을 Program이라 하고 이 變換作業을 Coding이라 한다. 따라서 Programming과 Coding은 區別된다.

〈指導 1〉 順序圖는 目的이 아니고 手段이며 반드시 必要한 것은 아니나 퍽 有用한 것이다.

Program을 Design하고 Documentation하는 데는 여러가지 方法이 있으나 그 중 順序圖는 가장 오래된 大衆化된 것으로 프로그램의 制御構造를 나타내는 것이 가장 큰 役割이다.

順序圖의 人氣는 점차 下落하고 있으나 프로그래밍 入門 코스에서는 매우 有用하다. 順序圖는 Debugging에 도움을 주는 등 長點도 많지만 短點도 있다. 順序圖와 프로그램 간의 變換은 그 表現이 唯一한 것만은 아니다. 順序圖는 ① Template라는 道具를 써서 프로그래머에 의하여 손으로, ② Flowcharting Program에 의하여 自動으로, ③ 부분적으로 ①, ②를 適用하여 半自動으로 그릴 수 있다.[3]

順序圖의 同意語나 用途에 따른 種類 및 變種



〈그림 1〉

도 많다. 高校에서 指導할 것은 Program Flowchart중 Detail Flowchart(詳細 順序圖)에 該當된다.

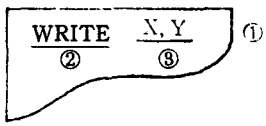
順序圖는 프로그래밍 言語에 따라 微細한 差異는 있지만 特定 프로그래밍 言語나 機種에 獨立的으로 普遍的이고 共通的인 事項만 指導한다.

順序圖는 空아닌 有限 個의 Node(또는 Box)들이 Directed Path로 連結된 Graph이다. 따라서 順序圖를 Walk나 有向 Network의 一種으로도 본다.[4),5)]

알고리즘의 2次元的 表現으로 Pictorial Format을 使用하여 Data에 行해지는 作用의 Sequence를 記述하는 그래프的 手段이다.[6)]

使用된 記號(Symbol)는 Outline 또는 Element라고 하며 그림 1의 5가지가 있다. Flowchart Program에서는 보통 ①~④를 Node라고 하지만 ②~④를 7)에서는 Node, 5)에서는 Vertex라고 한다. ②는 置換命令, ③은 比較, 分岐命令을 遂行한다. 위에서 각 Node의 Entry Point와 Exit Point를 調査해 보면 相異點을 알 수 있다.

順序圖에는 ① 規定된 記號에 ② 資料와 ③ 作用을 記述하는 말을 統合하여 表記한다. 例를 들면 그림 2와 같다.



〈그림 2〉

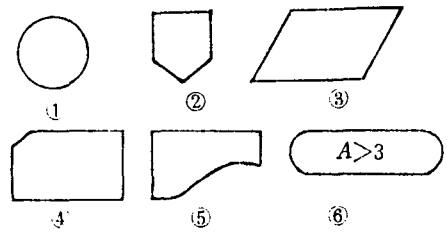
〈指導 2〉 現行 敎科書와 같이 몇개의 標準記號만을 使用하여라. 順序圖의 記號는 部分的으로 Informal한 것도 있지만 約定되어 있고 標準化되어 있다. 代表的인 것으로 ANSI와 ISO에서 採擇한 30여가지의 記號가 있다.[8)] 이들 대부분은 實務에 必要한 것으로 特殊處理나 入出力을 위한 것이다. 이 외에도 여러가지 體制의 標準記號들이 있다.[9)] 다만 詳細順序圖의 견지에서 다음 6~8가지를 결코 넘지 않아야 될 것이다.

(i) 前述한 그림 1의 5가지의 記號

그런데 보통 Collector는 省略하고 그리는 일이 많다. 그러나 論理의 흐름을 記述할 때 흐름선의 交叉를 避하여라. 以下 그림 3에서

(ii) Connector(連結子)를 써서 흐름선을 連結한다. ①, ②는 각각 on-page, off-page 連結子이다. 흐름선의 횟수를 줄이기 위하여 連結子를 使用하여라.

(iii) 入出力 共通記號 ③과 標準入出力記號 ④, ⑤를 使用하여라. 또 留意할 것은 ⑥과 같은 일부 책에 있는 判斷記號를 쓰는 일이 없도록 指導한다.



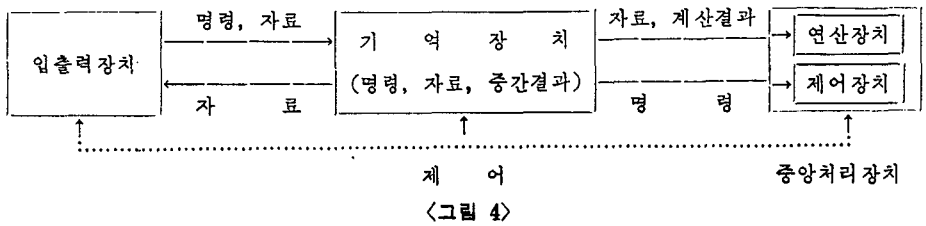
〈그림 3〉

3. Memory Cell과 Assignment

知識 Base와 推論 機構를 指向하는 제 5세대의 컴퓨터 시스템을 期待하지만[10)] 現在의 모든 컴퓨터는 Von Neumann Machine이다. 이 System의 基本構成은 〈그림 4〉와 같으며 그 特徵은 Stored Program Method이다.

프로그램은 記憶裝置에 貯藏되어 逐次的으로 實行된다. 따라서 論理의 흐름이 위에서 아래로, 左에서 右로 되게 順序圖를 그린다.

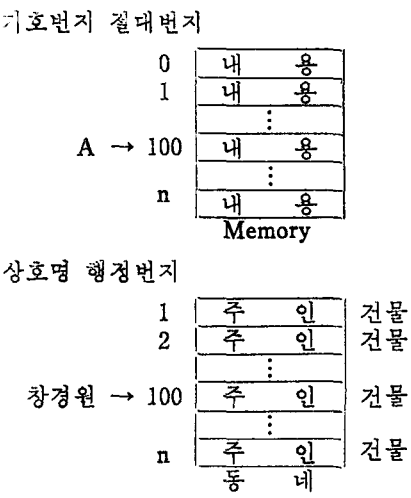
메모리의 構造, 資料의 表現, Addressing Mechanism등을 잘 알아야 되겠지만 간단히 代入文



〈그림 4〉

을 遂行할 수 있는 Logical Model을 생각하자.

〈指導 3〉 Memory의 構成을 그림 5와 같이 理解시킨다.



〈그림 5〉

메모리는 수많은 Cell(또는 Box)로 構成되어 있으며 각 Cell에는 Address(番地)가 指定되어 있고 항상 어떤 內容(命令이나 資料)이 들어 있다. 사람들은 建物을 찾을 때 行政番地보다 商號名을 찾듯이 대부분의 프로그래머는 記號番地라는 Name을 賦與한다. 따라서 Name은 그 Cell의 番地이름이다. Cell의 內容을 값(Value)이라고 한다. 그래서 Memory를 函數的으로 表現하면 다음과 같다.[11] Memory : Name→Value

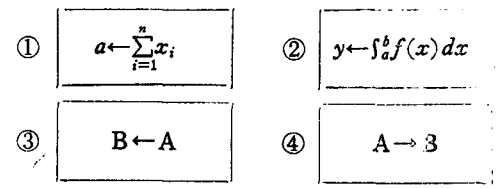
〈指導 4〉 變數는 알고리즘을 遂行하기 위한 Tool(그릇)으로 指導한다. 따라서 變數名은 이 그릇에 붙여진 이름이다. 주어진 問題를 풀기 위해서는 最小限 몇 種類의 그릇이 必要한가를 決定하고 이 그릇의 內容이 어떻게 變하는가를 考察한다.

Cell에는 여러가지 값이 代入될 수 있으므로 Variable이라고 볼 수 있다. 그래서 Name을 Variable Name이라고 한다. 그런데 컴퓨터는 스스로 프로그램에 나타낸 Name과 번지를 짝지워 記憶하고 있으며 必要時에 恒常 參照(Reference)할 수 있다.

컴퓨터는 Symbolic Manipulator 또는 Algorithm Processor라는 말보다도 具體的으로 무엇을 할 수 있는가를 理解시키는 것이 基本的인 先須槪念이다. 그중 必要한 것만 나열하면 다음과 같다.

① READ(外部媒體의 情報→入力裝置→記憶裝置)
② WRITE(記憶裝置의 內容→出力裝置→外部媒體에 表現)
③ MOVE(한 Cell의 內容을 다른 Cell로 옮긴다.)
④ COMPUTE(四則演算)
⑤ COMPARE(두 Cell의 記憶內容의 同一與否, 大小 比較判斷)

따라서 代入文의 右邊은 ① 四則演算, ② 累乘과 累乘根, ③ 其他 超越函數를 計算하는 內裝函數가 可能하지만 詳細順序圖에서는 〈그림 6〉의 ①과 ②처럼 그리지 않도록 指導한다.



〈그림 6〉

〈指導 5〉 Memory Cell은 錄音 Tape와 같은 特性을 지니고 있다고 指導한다.

A番地의 內容을 B番地에 옮긴다(MOVE)는 말은 Copy의 習慣化된 誤稱이다. (마치 數學에서 有理數라는 말처럼) Memory와 Tape의 다른 점은 全體的인 構造에서 前者는 電蓄 Disk와 같이 Random Access가 可能한 점이다. 위 경우를 順序圖로는 ③ 또는 ④와 같이 그리지만 대부분의 프로그래밍 言語를 보면 ③이 바람직하다. B←A는 다음 2動作으로 이루어진다.

① 먼저 A의 內容을 읽는다. 읽기 動作은 非破壞的이다. 이것은 錄音機의 Play動作과 같다.

② 다음에 B라는 番地를 찾아 그 Cell에 쓴다. 쓰기動作은 破壞的이다. 그래서 B←A후에 B의 값은 다시 回復할 수 없다.

이것을 프로그래밍 言語로 表現하면 實行文중의 代入文이 되는데 代入文의 右邊에 나타난 變數名은 그 變數名이 갖는 값이며, 左邊에는 반

드시 (하나의) 變數名이 나타나는데 이 때에는 그 番地를 찾아야 한다.

〈指導 6〉 BASIC, FORTRAN, PL/I에서 =는 等號가 아니고 代入演算子임을 强調하여야 한다.

BASIC, FORTRAN, PL/1등	B=A
ALGOL, PASCAL, ADA등	B : =A
APL	B←A
POP 2	A→B
COBOL	MOVE A TO B.

A=A+1에서 右邊의 A는 A라는 Cell에 記憶된 內容이고 左邊의 A는 A라는 Cell의 이름이다. 右邊에 記憶된 A의 過去의 內容이 이 代入文을 遂行하고 나면 左邊에 記憶된 A의 새로운 內容으로 되므로 A番地의 內容을 1增加시키는 셈이 된다.

4. 順序圖의 設計

〈指導 7〉 平易하고 簡明한 解法을 發見토록 指導한다.

現在 Software Engineering에서 重要한 것은 프로그램의 生產性으로 프로그램은 Readability와 Writability가 높아야 한다.

(i) 基本的이고 典型的인 模倣과 習得은 經驗的 理解에 도움이 되지만 固着된 觀念에 얽매이지 않도록 하여 多樣한 解法을 찾는 創造的인 면을 指導한다.

(ii) 特異하고 奇拔한 解法보다도 普通사람이 正常的인 思考로 찾아지는 平易한 解法을 求하는 生產的인 면을 指導한다. 從來의 프로그래밍이 技巧的인 觀念에 치우쳐 Programming is an art. 라고 하였지만 지금은 하나의 建物을 짓듯 Programming is an Engineering Descipline.으로 본다.[11]

(iii) 思考의 必然的인 結果로 解法이 誘導되도록 論理的인 면을 指導한다.

신동선 外 3名 : 수학 I (하) P.46의 보기 3에서 S=1+(1+2)+(1+2+3)+……+(1+2+3+ +100)을 풀어보자. 累加하는 項을 T라고 하면 T는 1, 1+2, 1+2+3, ……, 1+2+3+……+100, 따라서 S←S+T로 된다.

그런데 T를 調査해 보면

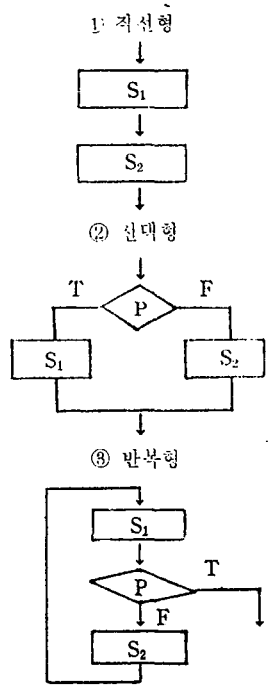
①, 1+②, 1+2+③, ……, 1+2+3+……+⑩ 여기서 T의 變化는 項의 順序에 해당되는 ○안의 數만큼씩 불어나므로 T←T+N, 물론 이때 N←N+1이다.

이것을 위 敎科書처럼 하는 것은 더 어려울 것 같으며 실제 설계에서도 그렇게 하지 않는다.

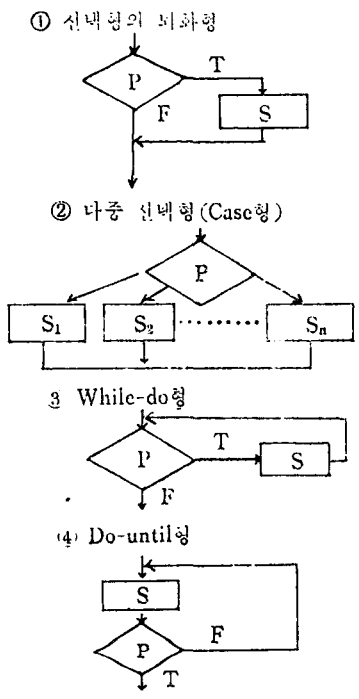
〈指導 8〉 構造化 프로그래밍의 견지에서 설계하도록 指導한다. 1970년대부터 Software에의 많은 硏究結果로 여러가지 프로그래밍 技法이 發見되었으며 構造文 프로그래밍은 그 중의 하나로 Chapin Charts나 Nassi-Schneiderman Charts라는 Structodiagram으로 表示할 수도 있다.[12]

現在 順序圖에 關한 國內書籍(3卷)이나 百여종에 이르는 國內의 각종 프로그래밍 册子, 그리고 商業系 高校등 學院에서 指導하는 프로그래밍 技法은 전혀 構造化 프로그래밍을 모르는 舊態依然한 繼續的 反復이다. 構造化順序圖는 그림 7의 3가지 基本型으로 되어 있다.

이것에는 그림 8의 4가지 擴張이 있다.[13]



〈그림 7〉



〈그림 8〉

따라서 3가지 基本型과 擴張의 組合(Case型은 可及的 삼가하고)만으로 設計하도록 指導한다.

〈指導 9〉 數列에 關한 指導에서 留意할 점은 다음과 같다.

(i) 初期値 賦與

System의 Power on과 더불어 대부분의 記憶裝置의 狀態는 不規則한 內容을 지니게 되므로 대다수의 프로그래밍 言語에서는 값이 指定되지 않는 Cell을 參照하면 Error가 된다.

Algorithm을 遂行하기 爲하여 論理的으로 必要한 것이 初期値이다. 例를 들면 空集合의 元素의 갯수는 0이듯이 어떤 Cell의 內容이 Clear 狀態로 되기 위해서는 數値的으로 0(文字값으로 空白)을 Set하여야 한다.

그런데 대부분의 級數에서 처음 몇개項은 不規則的인 것이 있을 수 있으며 規則的인 境遇에도 初項만은 바로 初期値로 賦與하는 것이 편리하다.

(ii) 一般項의 表示

數列을 漸化式으로 表現하기보다 數列이나 漸化式을 順序圖로 나타내기는 쉽다.

김연식外 1名 : 수학 I (상) P.162를 보면 導入過程에서 漸化式을 쓴 것은 無理이다. 이것은 Recursive Function의 槪念에 이르기 때문이다. 등차수열의 一般項을 A, 公差를 D, 合을 S, 項數를 N이라고 하면 一般項을 꼭 N으로 表示할 必要없이

$$A \leftarrow A+D,\ S \leftarrow S+A,\ N \leftarrow N+1$$

로 하면 된다. 이것은 바로 컴퓨터의 反復處理의 性質을 利用한 것이다.

끝으로 1981年 12月 31日字로 告示되어 1984學年度부터 實施豫定인 人門高等學校 數學科 敎育課程에는 수학 I 대수의 (자)項에 順序圖가 獨立的으로 設定되어 「① 順序圖의 規約, ② 順序圖에 의한 問題解法」으로 되어 있다.

References

1. Randal W. Jensen, Charles C. Tonies: *Software Engineering:* Prentice-Hall, Inc., 1979, p. 264.
2. Leon S. Levy: *Discrete Structures of Computer Science:* John Wiley & Sons, Inc., 1980, p. 209.
3. Boris Beizer: *Software Test Techniques:* Van Nostrand Reinhold Publishing, 1983, pp. 37-41.
4. Robert McNaughton: *Elementary Computability, Formal Languages, and Automata:* Prentice-Hall, Inc., 1982, pp. 58-60.
5. S.E. Goodman, S.T. Hedetniemi: *Introduction to the Design and Analysis of Algorithms:* McGraw-Hill Kogakusha, Ltd., 1977, pp. 19-26.
6. Ned Chapin: *Flowchart* (Encyclopedia of Computer Science and Engineering); Van Nostrand Reinhold Company; 2nd ed., 1983, pp. 633-641.
7. Rechard C. Linger, Harlan D. Mills, Bernard I. Witt: *Structured Programming:* Addison-Wesley Publishing Company, 1979, pp. 92-94.
8. Gordon B. Davis: *Introduction to Computers;* Mc Graw-Hill Kogakusha, Ltd., 3rd ed., 1977, pp. 88-104.
9. Roger G. Johnson: *Flowcharts and Decision Tables* (Programming Language Standardization); John Wiley & Sons, Inc., 1980, pp. 157-163.
10. 김한우 : 제 5세대 컴퓨터시스템의 시도(정보과학회지, 제 1권 1호) : 한국정보과학회, 1983. 3, pp. 45-49.

11. William A. Wulf, Mary Shaw, Paul N. Hilfinger, Lawrence Flon: *Fundamental Structures of Computer Science;* Addison-Wesley Publishing Company, Inc., 1981, pp. (iii), 182.
12. Hary Katzan, Jr.: *Invitation to Pascal;* Petrocelli Books, Inc., 1981, pp. 207-210.
13. Richard B. Hurley: *Decision Table in Software Engineering;* Van Nostrand Reinhold Company, Inc., 1983, pp. 129-131.