韓國軍事運營分析學會誌
第10卷, 第2號, 1984.12

HANSFOR : 한글 프로그래밍 언어

이 진 태*

Abstract

Hangeul programming language, written in Hangeul and having syntax of Korean, can enhance readability very much for its naturalness, and is suited for Hangeul data processing. But there exist several problems in implementing it due to the peculiarities of Korean such as verb position, variation of termination.

This paper proposes a solution for implementing Hangeul high level language and introduces a Hangeul programming language HANSFOR (Hangeul Structured Fortran), implemented according to the proposed method.

HANSFOR satisfies such requirements of Hangeul high level language as mentioned above and can handle Korean and Chinese character data efficiently. Especially, though it is based on Fortran, it is free-formatted and provides structured programming concept to compensate for defects of original Fortran. It is successfully implemented on NEC S/100.

1. 序 論

프로그래밍 언어의 목적은 프로그래머가 원하는 프로그램을 편리하게 작성할 수 있는 도구를 제공해 주는 것이다. 컴퓨터 사용자는 주로 프로그래밍 언어를 사용하여 컴퓨터에게 명령을 주게 되므로, 프로그래밍 언어에 대한 이해도가 곧 컴퓨터의 이해와 능률적인 사용을 가능하게 하는 중요한 요인이 된다.

그러나 기존의 프로그래밍 언어는 대부분 英語로 되어 있으므로 한국인에게는 어려움이 있으며, 만약 한국어로 된 프로그래밍 언어를 사용한다면 이미 한국어에 익숙해 있는 우리가 프로그래밍언어를 익히는데 필요한 노력은 반감되고, 독해 용이성의 증가로 인한 소프트웨어 생산성의 향상을 기대할 수 있다. 이미 일본을 비롯한 몇몇 나라에서는 自國語로 구성된 프로그래밍 언어를 개발하여 사용하고 있다. 최근 한글 데이타 사용량이 급증하고 한글 워드 프로세싱 시스템이 개발되는 추세로 보아서 한글 데이타 프로세싱 시스템의 개발은 당면 과제가 되었고, 한글 프로그래밍 언어의 개발은 이를 해결하는데 한 가지 좋은 방편이 될 것이다.

이 논문에서는 한국어의 특성을 고찰하고

* 한국과학기술원

적절한 한글 프로그래밍 언어를 설계하기 위해 고려해야 할 점들과 해결방안을 모색하였으며, 특히 한글 고급 언어 HANSFOR(Hangeul Structured Fortran)를 설계, 구현하고 사용하면서 느낀점과 개선점에 관하여 설명하였다.

HANSFOR는 한글로 작성하고 한국어 구문구조를 따르며, 한글·漢字 데이타를 사용할 수 있다. 특히 자유형식이며 스트럭춰드 프로그래밍을 할 수 있는 제어문장들을 제공하는 등 Fortran의 단점들을 보완하고 있다.

2. 한글 프로그래밍 언어

2.1 필요성

기존의 英語 프로그래밍 언어에서 英文 데이타의 처리는 간단하나 한글데이타는 英語코드로 풀어쓰거나 특별한 응용프로그램을 사용하여 처리한다. 한글 고급 언어를 사용하였을 때 얻어지는 이익들은 다음과 같다.

가. 비전문가들의 언어 이해와 습득이 용이하다.

나. 프로그램 코맨트와 단어를 한글이나 漢字로 기술하므로 읽기가 쉽고 모양이 아름답다.

다. 자연스러운 한국어 문장구조의 문장 사용으로 사고를 원활하게 할수 있고 오류를 줄일 수 있다.

라. 한글, 漢字 데이타를 직접 표현하므로 확인하기 쉽고 코드를 찾는 어려움이 없다.

마. 한글 프로그래밍 언어의 연구결과를기반으로 하여, 사용자와 컴퓨터의 의사소통에 사용되는 명령어를 비롯한 모든 언어체계를 한글 위주로 바꿀 수 있다.

바. 컴퓨터가 모든 사람에게 친숙해 짐으로 인력자원이 증대되고 한국어연구에도 자극제가 된다.

소프트웨어 가격이 점증하고 전산 전문 인력이 부족한 우리나라의 실정에 인력보강과 컴퓨터 국산화의 간접효과를 위해 한글 프로그래밍 언어의 개발은 유익하다.

2.2 한국어 문장구조의 특성과 문제점

이 절에서는 英語와 다른 한국어 구문구조의 특성을 살펴 보고, 그것이 파아싱과 프로그램 구조에 미치는 영향에 대하여 고찰한다.

(1) 英語의 최소단위는 알파벳이며,이 알파벳이 모여서 단어를 구성한다. 한글의 최소단위는 자음과 모음이며 이 자모가 조합되어 단음절을 형성하고, 단어는 이 단음절들로 구성된다. 이 단음절은 컴퓨터의 入出力 처리에 중요한 역할을 한다.

(2) 단어는 의미를 가진 최소단위로서, 한국어의 품사분류는 독특하다. 즉, 英語에서 전치사와 접속사가 중요한 품사이며, 다른 단어와 분리하여 쓰는 반면, 한국어에서는 조사와 서술사가 중요한 품사로 분류되며 조사와 어미는 각각 체언과 용언에 붙여 쓴다.

(3) 한국어는 우랄 알타이(Ural-Altaic)어족에 속하는 첨가어로서 관념어에 부속어가 첨가되는 형태를 취하며, 특히 부속어에 속하는 조사와 어미는 다양한 형태를갖는다.

특성 (3)과 특성(2)로 인하여, Lexical Analyzer는 조사와 어미를 분리, 분석하는 기능을 필요로 하게 된다. HANSFOR 에서는 Lexical Phase 이외에 Suffix Phase 를 두어 이 일을 처리하고 있다.

(4) 英語는 굴착어로서 술어가 문장의 앞부분에 나오는 left-to-right구조인데 비하여, 한국어는 중요한 술어가 문장의 뒷부분에 나오는 right-to-left의 구조를 갖

는다. 이 특성은 LR 파아싱에 불리한 조건이며 프로그램을 읽기 어렵게 한다.

프로그래밍 언어를 표현하는 문법이 LR(k) 문법이면 그 언어를 파아싱할 수 있는 DPDA(Deterministic Push-Down Automata)를 만들 수 있음이 증명되어 있다[9] 문법이 LR(k)라 함은 왼쪽에서 오른쪽으로 k개의 터미날 심볼을 조사하여 하나의 핸들을 가려낼 수 있음을 말하므로, 문장이 문장의 종류를 구별지어 주는 독특한 단어로 시작되는 것이 바람직하다.

2.3 해결방안

2절에서 살펴본 문제점들을 해결하는 방안으로 다음을 생각할 수 있다.

문제 (2)와 (3)을 해결하기 위해, Lexical Analysis에서 띄어 쓴 구문단위의 글자 문자열을 분리한 후 다시 체언에 붙은 조사나 용언에 붙은 어미를 분리하여 각각 중요부(root)와 부속부(ending)로 한다. 그리고 부속부는 분석하여 의미별 계급으로 분류한다.

문제 (4)를 해결하기 위하여서 몇가지 대안을 생각해 볼 수 있다.

첫째로, 문장끝의 주요어를 인식한 후 백트랙킹을 하는 파아싱 방법을 사용한다. 그러나 현재의 파아싱 이론에서는 백트랙킹을 허용하는 파아서는 부정되고 있으며 [6],내부에 다시 문장을 내포하는 것이 전혀 불가능하다.

둘째로, 한국어의 특성을 무시하고 英語처럼 동사를 문장 첫 머리에 둔다. 그러나 한국어는 체언, 용언의 순서로 異心構成을 이루고 있다는 言語學的 특성을 살리지 못하므로 한글 언어의 의미를 상실하게 된다.

셋째로, 동사대신 문장 첫머리에 올 수 있는 품사를 이용하여 문장의 종류를 판별하게 한다.

3장에서 설명될 HANSFOR는 셋째 방법을 택하여 문장 첫머리에 와도 자연스러운 부사, 명사를 최대한으로 이용하고, 문장 끝의 동사는 생략 가능하게 하여 읽기 쉬움(readability)과 능률성 중에서 선택할 수 있게 하였다.

3. HANSFOR의 설계

HANSFOR는 한국어 문장구조의 구문구조를 따르며, 한글로 작성하고, 英文 데이타 이외에 한글, 漢字, 특수문자 데이타를 자유롭게 사용한다.

HANSFOR프로그램은 Fortran언어로 전처리(preprocessing)하여 번역, 수행된다 그러나 자유형식이며, 스트럭춰드 프로그래밍이 가능한 제어문장을 제공하고, 보다편리한 코맨트 기술방식과 자유로운 라벨 사용을 허용하는 등 Fortran의 단점들을 거의 보완하고 있다. 이런 진보된 언어의 개념들은 사고내용을 프로그램으로 옮기기 쉽게 하며 좋은 프로그래밍의 습관을 붙이게 하므로 대중적 프로그래밍 언어의 구비요건이 된다.

3.1 자유형식과 코맨트

HANSFOR의 문장은 1컬럼에서 80컬럼 사이의 어느 곳에서나 시작할 수 있는 자유형식이다. 일반적으로 한 문장은 한 줄을 차지하므로 줄의 끝은 문장의 끝(;)을 암시한다. 그러나 쉼표(,)로 끝나는 문장은 자동적으로 다음 줄로 계속된다. 한줄에 여러 문장을 쓰고자 할 때는 각 문장끝에 문장끝 표시(;)를 하면 된다.

줄 가운데 어느 곳에나 샵(#)이 출현하면 코맨트의 시작으로 보고 그 줄의 끝까지는 무시한다. 따라서 문장의 뒷부분에 코맨트를 삽입할 수 있다. PL/I과 같이 코맨트의 시작과 끝을 명시하는 방법은 코맨트 삽입이 자유로운 반면, 부호의 누락으로 인한 오류유발의 가능성이 있다. 코맨트는 한글, 漢字, 특수부호 등이 모두 가능하다.

3.2 변수, 상수와 라벨

HANSFOR에서 알파벳은 1-바이트 코드 영숫자와 2-바이트 코드 문자의 집합이다. 변수는 이 알파벳의 문자열이므로 영숫자뿐아니라 한글, 漢字를 함께 사용할 수 있다. 원칙적으로 변수의 길이에는 제한을 두지 않았으나, 처음 12바이트(한글 6자)만 유효한 의미를 갖는다.

숫자상수(numerical constant)는 Fortran과 같으며, 문자형 데이타는 1-바이트 문자이외에 2-바이트 한글, 漢字, 그래픽 문자를 함께 사용한다. 논리상수로는 ".TRUE.", ".FALSE." 대신 "참", "거짓"을 사용한다.

assignment기호로는 APL에서와 같이 "<-"를 사용하여 assignment본래의 의미를 바르게 심어 주고, 릴레이셔날 오퍼레이터 "="와의 혼동을 피하게 하였다.

라벨은 문장 앞에 콜론(:)과 함께 쓰며, 숫자 대신 변수이름처럼 한글, 漢字, 영숫자를 사용하여 독해용이성을 높히게 하였다.

3.3 키워드

HANSFOR의 키워드는 의미가 상통하며 일반적으로 사용되는 우리말 용어로 바꾸었다. 그러나 아직까지 전산분야의 용어에 대한 통일된 우리말 용어가 없으므로, 서적과 주위의 의견을 참조하였으며, 더 적당한 용어로 대치할 수 있도록 테이블을 구성하였다.

주로 선언문이나 비수행문에 사용되는 HANSFOR의 키워드는 〈표1〉과 같다.

〈표1〉 **HANSFOR의 키워드**

HANSFOR	FORTRAN	HANSFOR	FORTRAN
프로그램	PROGRAM	형 식	FORMAT
서부루틴	SUBROUTINE	정수 〔들〕	INTEGER
함 수	FUNCTION	실수 〔들〕	REAL
배열변수 〔들〕	DIMENSION	논리수 〔들〕	LOGICAL
공통변수 〔들〕	COMMON	참	.TRUE.
동일변수 〔들〕	EQUIVALENCE	거 짓	.FALSE.
초기치 〔들〕	DATA	복수변수 〔들〕	COMPLEX
끝	END	정밀변수 〔들〕	DOUBLE-PRE

3.4 문장형식

HANSFOR의 문장은 크게 비수행문, 단순문(simple statement), 구조문(structured statement)의 3가지로 나누어지는데, 이 중에서 비수행문의 형식은 Fortran과 동일하며 〈표1〉과 같은 키워드를 사용한다.

스트럭춰드 프로그래밍은 이미 필수적이고 기본적인 소프트웨어 기술이 되었으므로, 대중적이고 교육적인 목표를 가진 프로그래밍 언어에는 구조문이 필요하다〔8〕.

비록 composition, IF-THEN-ELSE, WHILE-DO등3 종류의 구조문으로 모든 프로그램의 표현이 가능하다는 사실이 증명되어 있지만〔13〕, HANSFOR는 제어변수(control variable)가 있는 경우와 없는 경우에 GO-TO 문장이 없이도 불편없이 프로그래밍을 할 수 있도록 구조문들을 설계하였다.

EBN (Extended Backus Notation)형식으로 표현한 단순문과 구조문의 구문구조는 〈표2〉와 같다.

〈표 2〉 단순문과 구조문의 구문구조

문 장	EBN형 식
ASSIGNMENT	v = e ;
CALL	서브루틴 sub〔(〔a〔,b〕…〕)〕을〔수행하라〕;
CLOSE	닫음(olist) ;
FUNCTION	〔type〕함수fun(d〔,d〕…);
GOTO	문장 s으로 〔가라〕;
OPEN	엶(olist) ;
PROGRAM	프로그램 pgm;
READ	읽음(clist)〔iolist〕;
RETURN	돌아가라;
ST-FUNCTION	fun(〔d〔,d〕…〕)=e;
STOP	멈추어라;
SUBROUTINE	서부루틴 sub〔(d〔,d〕…)〕;
WRITE	씀(clist)〔iolist〕;
NO-ACTION	;
DO	반복하여 i=e1,e2〔,e3〕; st
EXIT	밖으로;
FOR	반복조건(s1 s2 e) 으로〔반복하여〕; st
FOREVER	영원히 〔반복하며〕;st
IF	만약 (e)〔이〕면 st〔아니면 st〕;
REPEAT	한계 (e)까지 〔반복하여〕; st
WHILE	조건 (e)동안 〔반복하여〕; st
COMPOUND	〔st〔,st〕…〕

(밑줄 그은 부분은 같은 종류의 조사나 어미변화가 가능함을 의미한다).

〈표2〉에서 보듯이 구조문은 단일문이나 문장군(statement group)을 수행할 수 있는데, 문장군은 한 쌍의 대괄호("〔" , "〕")로 싸인 일련의 문장들을 말한다. 이는 ALGOL의 begin.. end와 비슷하나 보다 간결하고 눈에 잘 띈다. 실제로 Ratfor에서는 중괄호를 사용한다〔8〕.

EXIT, FOREVER등의 문장은 루우핑의 창조나 탈출을 위해 REPEAT, WHILE 문장 못지 않게 중요한 역할을 한다〔11〕.

본래의 Fortran에서의 DO 문장은 제어변수가 양의 정수이어야 하며 단순 증가만 가능한 제한이 있으나, HANSFOR의 DO 문장은 이런 제한이 없이 자유롭고, 라벨을 사용하여 범위를 정하는 대신 바로 아래의 문장군에 대하여 반복 수행한다.

FOR문장은 제어변수의 초기치, 변화, 반복조건을 한꺼번에 기술하여 보다 일반적인 루우핑을 창조한다.

키워드는 될 수 있는 한 평범한 우리말을 사용하려 하였다. 예를들어 " STOP " 에 해당하는 "정지"보다 "멈추어라"를 택하였다.

3.5 조사와 어미의 변화와 생략

조사와 어미의 다양한 변화는 한국어의 중요한 특성이다.

HANSFOR는 어느 정도 다양한 조사와 어미 사용을 허용하여 자연스러운 문장이 되도록 하였다. 예를 들어서

만약 … 이면 문장 A로 가고

아니면 문장 N으로 가라

의 문장에서 보듯이 "로", "으로", "고" "라" 등을 사용함으로써 문장이 훨씬 자연스러워졌다.

또한, 생략하여도 뜻이 통하여 어귀나 조사, 어미는 생략가능하게 하여 코우딩의 노력을 덜게 하였다. 목적격 조사나 "하라"의 의미를 가진 어미의 생략은 흔히 사용되는 한국어 어법의 특성이다.

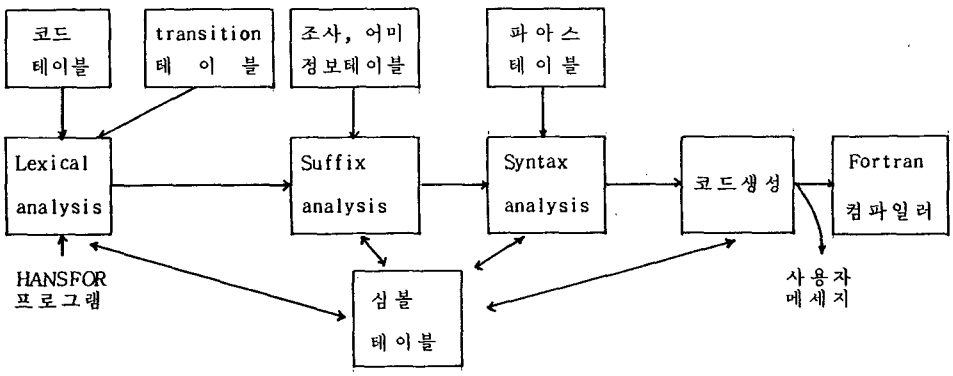
예) 서브루틴 SINE (X,Y)〔를〕 수행〔하라〕

4. HANSFOR의 설치

4.1 전체적구성

HANSFOR의 처리과정은 2패스로 이루어진다. 패스1에서는 주로 문장앞에 붙은 라벨을 조사하여 심볼테이블에 준비시켜 놓고 패스2에서 이용할 수 있게 한다.

패스2의 처리과정은 그림1과 같다.



〔그림1〕 패스2의 처리과정

4.2 Lexical Phase

Lexical Analyzer는 HANSFOR프로그램을 한번에 한 바이트씩 읽어 들여서 이름, 상수, 오페레이터등의 토큰들을 出力한다.또한 코맨트 부분을 제거하고 줄의 끝을 문장의 끝(;)으로 인식할 것인가 말것인가를 결정한다.

이 시스템에서의 알파벳은 2-바이트 문자까지 고려하여 다음과 같이 정의된다.

α = { 1-바이트코드 영숫자 문자 }

$\beta 1$ = { 2-바이트 문자의 첫번째 바이트 }

$\beta 2$ = { 2-바이트 문자의 두번째 바이트 }

$\triangle$ = 2-바이트 공백문자

단, ($\alpha \cap \beta 1 = \phi$)

알파벳 $\Sigma = \{ x \mid x \in \alpha$ 또는 $x \in \beta 1\ \beta 2, x \neq \triangle \}$

문자열 $\Gamma = \{ s \mid s \in \Sigma^* \}$

4.3 Suffix Phase

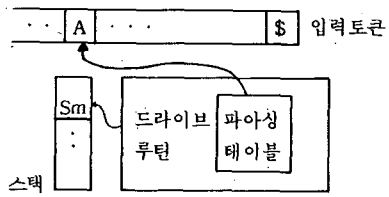
지나치게 다양하거나 言語學的인 조사,어미의 사용은 컴파일러의 능률을 저하시키므로 HANSFOR에서는 다음과 같은 4종류의 조사, 어미로 분석한다.

계급	보 기	특 성
0	을, 를, 들, 하라	목적격조사,종결성어미
	하고, 라, 고, 어라	복수형접미사 생략가능
1	에서, 서	처소격조사
2	으로, 로	방향격조사
3	하며,하여, 며, 여	부사형어미

계급이 "이름"인 토큰 문자열을 분석하여 중요부(root)와 부속부(ending)로 분리할 때, 중요부는 길이가 길고 자유분방한 형태이므로, 스택을 사용하여 뒤에서 앞으로 비교해야 한다.

4.4 Syntax Phase

HANSFOR에서 사용하는 파아싱 방법은 LALR(Look Ahead LR)파아싱 방법이다 (그림2).



〔그림 2〕 LALR 파아싱

LALR 파아싱은 강력하면서 canonical LR 파아싱에 비해 테이블이 크지 않은 장점이 있다[6]

4.5 코드생성

코드생성 Phase에서는 Syntax analyzer에서 보낸 상태와 토큰에 따라 수시로 적절한 Fortran중간코드를 발생한다. 이때 생성되는 Fortran 원시 프로그램은 프로그램 구조에 따라 적절한 *indentation*을 하여 읽기 쉽게 하였다.

5. 실험 및 결과

이 장에서는 HANSFOR기능의 일부를 사용한 실험 프로그램의 견본과 그 결과를 제시한다. 선택된 예는 비교적 간단한 알고리즘의 프로그램이지만 英文 프로그램에 비하여 자연성 (naturalness)과 읽기쉬움 (readability)이 우수하며, 한글·漢文 데이타 처리와 잘 조화됨을 볼 수 있다(그림 3).

그림 3의 프로그램에 그림 4와같은 입력 데이타를 넣으면 그림 5와 같은 출력 결과를 얻는다.

```
1982 11
1982 12
```

〔그림 4〕 입력데이타

1982年 11月

일	월	화	수	목	금	토
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

1982年 12月

일	월	화	수	목	금	토
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

〔그림 5〕 출력결과

```
    프로그램  CALEND
#  이것은 달력을 인쇄하는
#  HANSFOR 표본 프로그램입니다.
#      入力 : 년도 , 월
#      出力 : 한달치 달력
정 수 들  년도, 월, 화면, 프린터, 날짜, 첫날요일, 날짜끝
배열변수들  연습장(7), 숫자들(31)
조 기 치  숫자들/ ' 1', ' 2', ' 3', ' 4', ' 5', ' 6', ' 7',,
                  ' 8', ' 9', '10', '11', '12', '13', '14',,
                  '15', '16', '17', '18', '19', '20', '21',,
                  '22', '23', '24', '25', '26', '27', '28',,
                  '29', '30', '31'/
초기치들  화면, 프린터/1, 2/, 공란/' '/
초기치들  밑줄시작, 밑줄끝, 옆줄/'  ', '  ', '  '/  #특수문자
    염(화면)
    염(프린터)
入力 : 읽음(화면, 형식1) 년도, 월
    형식1 : 형식(5뛈, 정4, 5뛈, 정2)
          # 첫날요일과 날짜길이를 계산한다.
計算 : 서브루틴 MONTH(년도, 월, 첫날요일, 날짜끝)를 수행하라
          # 머릿글과 날짜들 인쇄
出力 : 씀(프린터, 형식2) 년도, 월, 밑줄시작, 밑줄끝, 옆줄, 옆줄
    형식2 : 형식(10뛈, 정4, '年', 정2, '月'/,
                4뛈, 글2, 22뛈, 글2/,
                4뛈, 글2' 일월화수목금토', 글2, ' ')
          # 날짜들을 인쇄한다.
    날짜 < - 1
    한계 (날짜 > 날짜끝)까지 반복하여
      〔 I < - 1
        조건(I < = 7)동안  # 일주일 날짜를 채우고
          〔만약 (I < 첫날요일 & 날짜 = 1 | 날짜 > 날짜끝)이면
                연습장(I) < - 공란
           아니면
                〔연습장(I) < - 숫자(날짜)
                 날짜 < - 날짜 + 1
                〕
           I < - I + 1
          〕                          # 한줄씩 인쇄한다.
    씀(프린터, 형식3) (옆줄, (연습장(J), J < - 1, 7), 옆줄)
    형식3 : 형식(4뛈, 글2, 1뛈, 7(글2, 1뛈), 글2, ' ')
    〕
    씀(프린터, 형식4) 밑줄시작, 옆줄, 밑줄끝, 옆줄
    형식4 : 형식(4뛈, 글2, 22뛈, 글2, 글2, ' ')
    닫음(화면)
    닫음(프린터)
    멈추어라
    끝
```

〔그림 3〕 표본 HANSFOR 프로그램

6. 結 論

본 논문에서 얻은 주요 結論들을 요약하면 다음과 같다.

가. 한글 데이타 사용량이 증가되고, 프로그램의 유지보수를 위한 소프트웨어 가격이 중요하여질 수록 한글 프로그래밍 언어의 효과는 증대된다.

나. 한국어의 특성으로 인해, 동사 이외에 부사, 명사의 적절한 이용으로 파아싱이 원활한 한글 프로그래밍 언어의 설계가 가능하다.

다. 한국어는 조사와 어미의 활용에 따라 문장의 의미가 좌우되므로 체언과 용언에 붙은 조사와 어미를 분리, 분석하는 단계가 필요하며, 스택을 이용한 능률적 분석이 가능하다.

라. 모델로 제작된 한글 고급 언어 HANS-FOR는 상기한 요구조건을 잘 만족시키며, 英文 프로그래밍 언어에 비해 월등한 자연성과 읽기쉬움을 보여주며, 한글 데이타 처리에 효율적이다.

마. 한글 프로그래밍언어의 최종목표는 한글 고급언어를 위한 하드웨어 개발이며, 이를 위해서는 한글 언어 전처리 단계, 한글 언어 직접 번역단계를 거쳐야 한다.

참 고 문 헌

〔1〕 李鎭泰, "한글 프로그래밍 언어의 설계 및 구현," M.S. thesis, KAIST, (1983).

〔2〕 趙廷完, "컴퓨터를 위한 한글 고급언어의 특성과 이를 위한 최적시스템에 관한 연구. 연구보고서, KAIST, (1982).

〔3〕 김민수, "국어문법론," 일조각, (1971).

〔4〕 성광수, "국어조사에 대한 연구," 형설출판사, (1987).

5. Kernighan, B. W., "Software Tools," Addison-Wesley, (1976).
6. Aho, A. V. and Ulman., J. D., "Principles of Compiler Design," Addison-Wesley, 1977).
7. Hoare, C. A. R., "Hints on Programming Language Design," IEEE, (1980).
8. Knuth, D. E., "Structured Programming with GO-TO statement," Computing Survey, vol. 6, no. 4 (1974), pp. 261-301.
9. Knuth, D. E., "On the Translation of Language from Left to Right," Info. and Contr. vol. 8, (1965). pp. 607-639.
10. Pratt, T. W., "Programming Language: Design and Implementation," Prentice-Hall, Inc., (1975).
11. Peterson, W. W., "On the Capability of While Repeat, and Exit Statements," CACM, vol.16, no. 8, (1973), pp. 503-512.
12. Aho, A. V. and Ullman, J. D., "The Theory of Parsing, Translation, and Compiling," vol. I, II, Prentice-Hall, (1973).
13. Bohm, C. and Jacopini, G., "Flow-diagrams, Turing Machines, and Languages with only Two Formation Rules," CACM, vol. 9, no. 5, (May 1966), pp. 366-371.
14. Aho, A. V. and Johnson, S. C., "LR Parsing," Computing Surveys, vol. 6, no. 2 (June 1974), pp. 99-124.