초등학생들의 논리적 사고력 신장을 위한 선언적 프로그래밍의 교육

김윤식, 한선관
경인교육대학교 대학원 컴퓨터교육학과
k24u@hanmir.com, han@gin.ac.kr

요 약

본 연구는 초등학생들의 논리적 사고력을 신장시키기 위해 지식 기반 프로그램인 선언적 프로그램을 통해 교육현장에서도 적용할 수 있는 프로그래밍 교육을 제언하고자 한다. 학생들에게 논리적 사고 중에서도 협의의 논리적 사고 즉, 기호적 사고, 분석적 사고, 추론적 사고, 종합적 사고를 분석적 방법을 통해 실제 프로그래밍을 해 봄으로써 연역적 사고 또는 귀납적 사고를 보다 효과적이고 체계적인 프로그래밍을 할 수 있도록 지도함으로써 제 8차 교육과정에서의 컴퓨터 교육과정의 일부분으로서의 프로그래밍의 마인드를 제시하였다. 따라서 본 연구는 선언적 프로그램을 통해서 초등학교 학생들의 논리적 사고력 신장를 위하여 프로그래밍 교수학습의 방법적인 측면을 제시하고자 한다.

1. 서론

모든 분야에서 컴퓨터가 중요한 역할을 담당하는 정보화 사회로 접어들었다. 지금까지의 초등학교 컴퓨터 교육은 단지 워드나 엑셀, 파워포인트, 인터넷 등의 컴퓨터 활용교육에 치우쳐 왔다. 실제 컴퓨터 교육을 통해서 학생들의 사고력 향상을 돕는다거나 문제해결력을 길러준다는 교육과는 본질적으로 서원한 면이 적지 않았다. 컴퓨터 소양을 기르고 컴퓨터 활용 교육을 확산함으로써 정보화사회에 적절히 대응할 수 있는 자질을 기르기 위한 컴퓨터 교육의 일반적인 목표는 세 가지를 들 수 있는데, 그 중 논리적 사고력과 문제 해결력을 기르는 목표를 달성하는 데 있어서 유용한 교육적 도구의 하나는 교육용 언어를 통한 프로그래밍이다[1]. 문제해결의 한 유형으로서, 프로그래밍의 중요성은 프로그래밍 언어 습득보다는 프로그래밍을 통해서 학습자의 고등인지기술을 향상시킬 수 있다는 것에 있다. 즉, 코딩 과정에서는 논리적 사고력을, 오류 검증 및 수정작업에서는 반성적 사고 능력을 향상시킬 수 있다. 새로운 교육용 프로그래밍 언어는 간결하고 학습자와의 뛰어난 상호작용성을 지니고 있어야 할 뿐만 아니라 논리적 추론 기능을 갖추어야 한다. 그러므로 본 연구에서는 이론적 고찰을 통해 논리적 사고와 프로그래밍과 교육용 언어의 교육적 가치를 확인하고 논리적 사고력 향상을 위한 사고력 검사의 요인분석 및 지도방안을 설계하고자 한다.

2. 이론적 배경

2.1. 비판적 사고력

비판적 사고 개념과 아주 밀접히 연관되어 있는 논리적 사고와 창의적 사고의

개념은 비교적 우리에게 익숙한 개념이다. 그러나 이 논리적 사고와 창의적 사고 개념 역시도 그 적용 범위가 하나로 확정되어 있는 것이 아니다. 즉, 표준의 의미 이외에도 넓은 의미와 좁은 의미의 논리적 사고와 창의적 사고 개념이 존재한다는 것이다[2].

1) 연연적 사고 방법

연역적 사고는 전체에 대한 지식이나 일반적인 법칙 또는 원리에서 출발하여 부분에 관한 지식이나 특수 사례 등을 이끌어 내는 방법이다. 즉, 공리, 정의 또는 이미 증명된 정리로부터 논리에서의 유효한 추론에 의하여 새로운 정리를 유도하는 방법을 말한다. 이와 같은 사고는 대개 한번에 한 단계씩 진행한다. 그 단계는 명백히 드러나며, 분석적으로 사고하는 사람은 그 단계를 타인에게 충분히 설명할 수 있다. 또한 자기가 어떤 내용의 지식을 다루며 어떤 방법으로 그 지식을 다루는가를 비교적 명확히 알고 있다.

2) 귀납적 사고 방법

귀납적 사고의 방법은 몇몇의 예를 설명하면서 일반화를 추출하는데 도달하기 위한 발견술(heuristic)적인 방법으로서 상당한 직관성을 내포하고 있다. Husserl에 의하면 모든 지식은 의도성과 직관의 산물이다. 직관이 배제된 지식이란 없으며 더욱이 귀납은 본질적으로 경험에 속하고 경험과 분리될 수 없으며 경험 그 자체이기 때문이다[3].

3) 유비 또는 유추적 사고

인간은 미지의 개념과 추상적 개념을 이미 알고 있는 개념과 결부시켜 생각하므로써 개념의 형성뿐만 아니라 문제해결 과정에 크게 개입한다고 볼 수 있다. 예를 들어 미지수가 두 개인 일차연립방정식의 해의 존재를 이해하는 데 좌표평면 위에 직선을 도입하여 이해하는 방법이라든지, 두 직선과 만나는 평행선들 사이 길이의 비를 구하는 데 삼각형의 닮음을 이용하는 것 등 유츄는 문제해결에 있어서 일반적으로 행해지는 사고 방법이라 할 수 있다.

4) 직관적 사고 방법

직관적 사고는 형식적 증명과 같은 외재적 정당화의 요구 없이 자명하고 즉각적으로 받아들여지는 인지 형태라고 볼 수 있다. 사실 직관은 추정, 선택, 전체성의 복잡한 메카니즘을 기초로 한다. 즉 제한된 양의 정보로부터 간접적인 외삽을 가하거나, 조화되지 않는 단서를 제기하고 단일하고 완전한 의미에서 준거하여 조직화 하는 선택 과정을 통해서 전체적인 시각을 제공해 준다. 직관은 무의식적 활동이고 내적으로 일관된 것만을 자동적으로 인지하는 것이다.

Brunner에 따르면 직관적인 사고는 지식의 구조를 지식의 구조를 기초로 가능하다. 이 구조가 지식의 단계적 발달이 아닌 사고의 도약을 가능하게 한다. 지식의 구조상으로 보아 어떤 것들이 서로 관련되어 있는가를 이미 알고 있을 때, 몇 가지 소수의 단서를 사용하여 유의미한 결론을 내리는 것이다. 따라서 사고 내용과 관련된 경험과 지식을 직관적 사고의 필요조건이다. 교육에서 직관적인 사고의 중요성은 이미 인식되어 있다. 교육내용이 분석적 방법으로 제시되기 전에 직관적으로 이해되도록 가르쳐야

하며, 그 두 종류의 사고가 상보성을 발휘함으로써 교육 효과를 높일 수 있다고 하는 많은 주장이 있다[4].

2.2. 프로그래밍 학습과 사고

1) 프로그래밍과 사고 기능

프로그래밍이 사고를 자발적으로 훈련시킬 수 잇다는 신념에 공헌하는 두 가지 주요한 요인이 되는 것으로는, 인공지능과 Piaget의 동화의 개념이 있다.

(1) 인공지능의 개념

인간의 인지의 복잡성을 본딴 프로그램을 구성한다는 것은 그러한 행동을 이해하는 방법으로 볼 수 있다. 우리는 컴퓨터가 무엇인가를 하도록 분명하게 가르침으로써, 사고에 관하여 더 많이 배운다는 것이다. 프로그래밍하는 학생들은 그들의 가정을 상세화하고 그들의 문제해결 방법의 단계를 정확하게 구체화하기 때문에, 프로그래밍의 분명한 속성상 반드시 문제해결 과정에 관하여 학습하는 것으로 볼 수 있다[5].

(2) Piaget의 동화의 개념

Papert(1980)는 자발적인 문제해결 경험을 통한 Piaget식의 지식의 획들 근거에 대한 열렬한 옹호자였으며, "계획적인 혹은 조직된 교수없이 발생하는 어떤 과정"에서의 "교육과정 없는" 학습이 프로그래밍 학습의 효과라는 생각에 광범하게 영향을 미쳐 왔다[6].

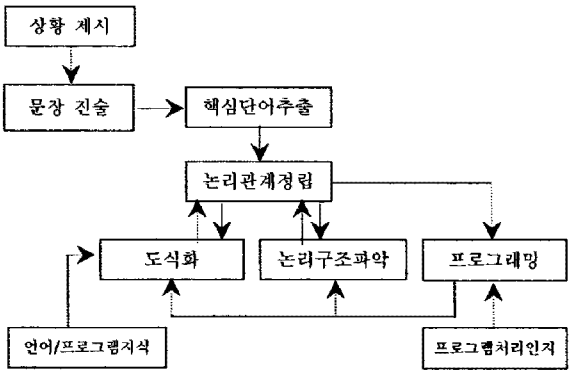
2) 프로그래밍과 계획 과정

사고에 관한 프로그래밍의 긍정적인 효과들에 관한 주장들 중의 하나가 계획의 영역에 있어 왔다[7]. 프로그래밍 경험은 "발견술", 즉 계획, 관련된 문제의 찾기, 혹은 문제를 부분으로 나누어 해결하기 등과 같은 어떤 영역에서 문제해결을 위하여 유용한 문제들에의 명료한 접근 방법들의 발견술을 더욱 촉진시키는 결과를 초래할 것이다. 프로그래밍의 합리적 분석과 성인 프로그래머들의 관찰의 결과, 계획이 프로그래밍에 있어서 중요한 방법임이 자명한 것으로 나타났다. 전문가의 수행 검사 결과 프로그래밍 문제가 형성되면, 프로그래머는 프로그래밍 부호로 쓰여지게 될 프로그램 계획이나 설계를 상세히 배치한다. 전문 프로그래머들은 많은 시간으로 프로그램 설계 계획에 소모하며, 문제 분해, 부분 목표의 설정, 알려진 해의 재생, 관련 프로그램들로부터 유사한 부초의 변경, 그리고 평가 분석과 프로그램 구성 요소들의 오류 분석과 같은 유용한 많은 전략들을 지니고 있다[8].

3) 문제해결력

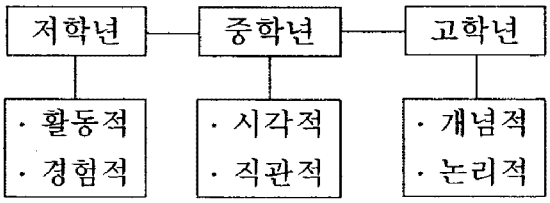
문제란 바라는 상태는 있으나 그것에 대한 명백한 해결점이 보이지 않을 때 발생한다. 따라서 문제를 해결한다는 것을 주어진 초기상태(Initial state)를 바라는 최종상태(Goal state)로 바꾼다는 의미이다. 결국 문제해결은 문제 공간에서 초기 상태로부터 최종 상태로 가는 최적의 길을 찾는 과정이라고 할 수 있으며 결과보다 일련의 과정을 더 중요시하는 개념이라 하겠다. 문제해결력은 문제를 해결하는 능력으로서 분류학에서 말하는 고등전신 기능이라 할 수 있다.

Newell과 Simon의 정보처리이론에 따른 문제해결 과정을 다음과 같이 재구성하였다[9].



[그림1] 정보처리이론 문제해결과정모형 재구성

여러 학자들의 주장을 종합하여, 문제 해결 과정은 거의 공통적으로 주어진 문제를 파악하고 문제해결 계획을 수립하여 계획을 실행에 옮기고 난 뒤 반성, 검증의 절차를 거치고 있다고 하였다. 그리고 문제 해결의 학습 지도에서 학년별로 [표1]과 같은 기본적인 지도 관점에 따라서 지도하려는 것이 공통된 특징이라고 하였다.



[그림2] 문제 해결 학습 학년별 지도 관점

3. 논리적 사고력 향상을 위한 사고력 검사의 요인분석 및 지도방안 설계

3.1. 사고력의 검사의 하위 요인 분석

[표 1] 사고력의 검사 하위 요인 분석

분석적 사고: 이해·분석력	◎개념적 이해 -단어 유비, 문장 전화 직접 추리 -선결문제 해소 -결론 찾기, 근거 찾기, 논증 찾기, 구조 분석 -느슨하거나 자명한 추리를 이용하여 바꾸어 답하기, 비일관적, 합치, 양립가능, 함축찾기
논증적 사고: 추론·논증력	◎연역(명시적 지식의 산출) -결론도출:삼단논법 -구조평가, 종합평가 -선언지 긍정의 오류 ◎귀납(확장적 지식의 산출) -귀납적 일반화, 가설 추리, 유비 추리, 인과 추리 -생략된 전제 찾기 -유사 형태 찾기 -잘못된 유추, 근시안적 귀납, 성급한 일반화, 인과 혼동
변증적 사고: 종합·대안력	◎논리퍼즐 게임(연역적 문제해결) -순서정하기 -길 찾기 -짝짓기 -그룹 짓기 -시간표 짜기 -행렬 논리 -다이어그램 -자기지시 ◎상황추리(귀납적) -복잡한 상황에서 배경 지식을 활용한 추리 -의사 결정/귀납적 문제해결 -상당히 이질적이거나 복잡한 상화의 유사 형태 찾기 -교훈 찾기 ◎발상 전화(재정의) -발상전환적 문제 해결 -발상전환적 논증·평가 -발상전환의 유사 형태 찾기

3.2. 논리적 사고 향상을 위한 지도 방안-연역과 귀납논리 중심으로

3.2.1. 저학년을 대상으로 한 언어의 기호화 지도방안

다음은 저학년 학생을 대상으로 한 언어의 기호화 방안인데, 저학년은 활동적이고 경험적인 교육적 활동을 통해 가역적 사고와 논리적 사고를 강화 시켜줄 수 있다. 간단한 문장을 중심으로 기호화를 한다면 학생들은 보다 쉽게 접근할 수 있는 데, 저학년 학생들에게는 규칙이나 사실에 대한 <약속하기> 과정이 필요하다. 왜냐하면 여러 단어를 기호화

하다보면 저학년 학생들의 집중력과 암기력이 떨어지기 때문이다.

예) 만일 배고프다면, 너는 밥을 먹을 것이다.

예) 만일 춥다면, 너는 옷을 입어야 한다.

[표 2] 저학년 약속하기 단계

약속하기	
만일(고정)	■
배고프다(변함)	▲
너(고정)	□
밥을 먹다(변함)	△
▲(원인)→△(결과)	

저학년 학생들은 다음과 같은 기호화 단계를 반복하게 하며, 여기에 다른 사실과 문장의 관계어와 사실들을 첨가할 수 있다.

첫 번째 과정 : ■ ▲, □ △

두 번째 과정 : ■ ▲, □ ▽△

3.2.2. 고학년을 대상으로 한 언어의 기호화 지도방안

고학년은 개념적이고 논리적인 사고가 가능하기 때문에 일반적인 문장을 제시하고 간단한 영문 표기식의 기호화 과정을 필요하다. 여기에서도 또한 저학년의 단계처럼 <약속하기> 과정이 필요하다. 왜냐하면, 다양한 문장과 연역적 논증이나 귀납적 논증을 일반화된 기호로 나타내기 위해서는 서로의 규칙의 동일성이 필요하기 때문이다. 하지만 처음 지도단계에서는 학생들의 발산적 사고가 활성화 되도록 임의적인 기호화 과정 또한 필요하다.

다음은 고학년 학생들에게 지도과정의 예이다.

예) 만약 철수가 정직한 하다면 너의 물건을 훔치는 일은 하지 않을 것이다.

[표 3] 고학년 약속하기 단계

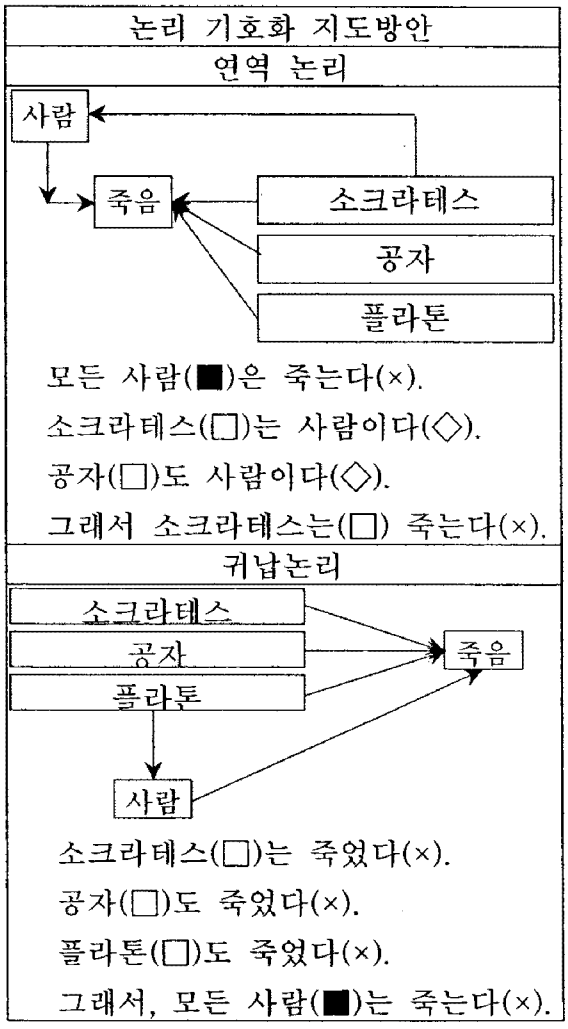
약속하기	
만일(고정)	if
~하다면	:
철수	culsu
정직하다.	▲
너의 물건	you
훔치지 않는다	△
▲(원인)→△(결과)	

첫 번째 과정 : if culsu ▲ : you △

두 번째 과정 : if culsu ▲ : you □ △

3.3. 기호의 도식화 및 프로그래밍 단계

[표 4] 도식화 단계 및 기호의 논리구조



위의 [표4]는 초등학교 학생들에게 논리적 구조의 파악 능력을 한층 뛰어나게 할 뿐만 아니라 기호화를 통한 프로그래밍 사고에 한 단계 접근이 용이하다. 막연한 프로그래밍을 교육하면 함수가 무엇인지, 메서드나 속성의 다양한 프로그래밍의 언어를 습득해야하기 때문에 학생들은 어려움을 겪게 된다. 하지만 도식화와 기호화를 적절히 사용한다면 다른 시각에서 프로그래밍 언어에 접근할 수 있다.

3.4. 기호화의 프로그램밍 단계

[표 5] 언어논리와 기호화의 관계

인물 : socrates(□)	죽는다 : mortal(×)
사람 : person(◇)	모든 : X-(■)
모든 학생들이 영어를 잘 하는 것은 아니기 때문에 논리적 체계를 잡기 위해서는 한글과 영어를 혼용해서 사용하게 한다. 하지만 프로그래밍할 때는 영어만을 사용하게 한다.	

-연역

모든 사람(■)은 죽는다(×).
소크라테스(□)는 사람이다(◇).
공자(□)도 사람이다(◇).
그래서 소크라테스는(□) 죽는다(×).
=>mortal(X) :- person(X).
=>person(socrates).

-귀납

소크라테스(□)는 죽었다(×).
공자(□)도 죽었다(×).
플라톤(□)도 죽었다(×).
그래서, 모든 사람(■)는 죽는다(×).
=>person(X) :- mortal(X).
=>person(socrates).

이러한 과정을 통해서 언어적 논리를 도식화하고 논리들의 관계성을 파악한다. 그리고 이 관계성을 기호화해서 하나의 프로그래밍화 한다. 다음은 기호화의 프로그래밍된 몇 가지의 예이다.

예1)모든 사람은 죽는다.

?-(mortal(X) :- person(X).

예2)보스톤에 어떤 소비자가 살고있는데, 그들의 신용등급은 몇입니까?
(What customers live in Boston, and what is their credit rating?)

?- customer(X, boston, Y).

예3)챕터 2의 주제는 무엇입니까?
(What is the title of chapter 2?)

?- chapter(2,Title).

예4)윈도우의 메인 좌표는 얼마입니까?
(What are the coordinates of window main?)

?- window(main,Row1,Col1,Row2,Col2).

3.5. 프로그램밍의 추론화 단계

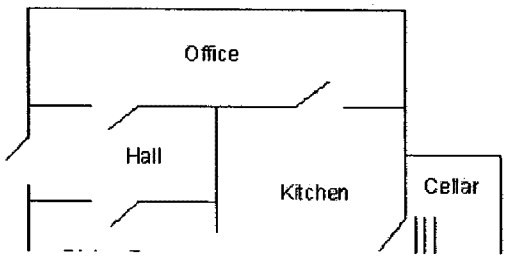
[표 6] 나니 검색 프로그램밍 소스

Nani Search(나니 검색)

//방의 종류
room(kitchen).
room(office).
room(hall).
room('dining room').
room(cellar).
 	
//door은 2개의 방이 연결된다.
//그래서 문에는 2개의 변수가 존재한다.
door(office, hall).
door(kitchen, office).
door(hall, 'dining room').
door(kitchen, cellar).
door('dining room', kitchen).

```
//방에 있는 물건
location(desk, office).
location(apple, kitchen).
location(flashlight, desk).
location('washing machine', cellar).
location(nani, 'washing machine').
location(broccoli, kitchen).
location(crackers, kitchen).
location(computer, office).
//먹을 수는 종류
edible(apple).
edible(crackers).
tastes_yucky(broccoli).
here(kitchen).
//손전등의 처음 상태와 위치 지정.
turned_off(flashlight).
here(kitchen).
```

학생들은 위의 상황에 맞게 그림을 그리며 논리적인 구조를 짜낸다.



[그림 3] 논리의 도식화된 결과

논리적인 확립을 위해서는 첫째, 도식화가 필요하다. 고학년 학생들은 발달단계에 맞게 기호화나 상징화를 할 수 있는 단계로서, 복잡한 구조일수록 도식화를 통해 정확한 정보를 얻을 수 있다. 둘째, 학생들에게 "만일 너희가 컴퓨터라면 너희는 어떤 방법으로 정보를 처리할 것인가?"를 물어보고 그 단계를 삼단 논법이나 관계성을 파악할 수 있도록 한다. 셋째, 논리적인 언어 구조로써의 접근에서 프로그래밍적 접근을 통해 학생들은 생소한 컴퓨터 언어를 배우는 것보다는 쉽게 프로그래밍 언어의 접근의 용이성과 분석적 사고력 향상을 꾀할 수 있을 것이다.

교수학습과정 동안에 다음의 단계가 이루어진다.

(1) 집의 구조와 방의 위치를 정하게 한다.

(2) 방과 방 사이에는 문이 존재하는 데, 문의 위치를 그리게 한다.

(3) 각 방에는 물건이 존재하는 데, 물건의 위치를 지정하게 한다.

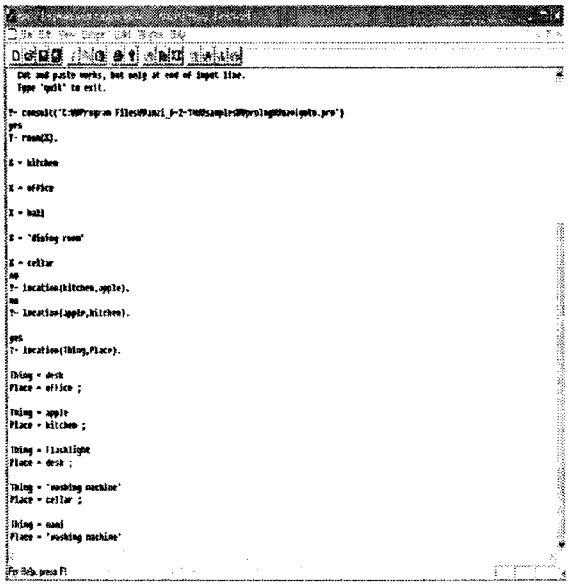
(4) 도식화된 그림과 기호를 통한 한글과 영어를 혼용하여 문장을 만들게 하고, 공통된 단어를 정의해 주며 프로그램을 짜기위한 기본 단어와 사실들을 구조화 시킨다.

(5) 이런 과정을 통해 학생들은 다음과 같은 프로그램을 짜게 하고 기호화된 언어를 프로그래밍하도록 한다.

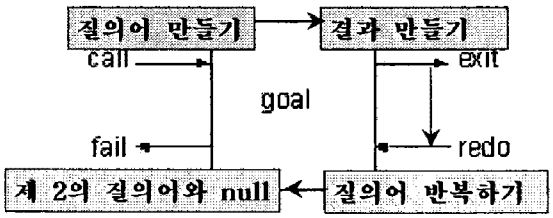
[표 7] 나니 검색 프로그램밍 소스

```
//부엌에는 어떤 것들이 있습니까?
?- location(X, kitchen).
//호출하기
//부엌에 있는 물건 불러오기
CALL: - location(X, kitchen)
//부엌에는 사과가 있다.
EXIT:(2) location(apple, kitchen)
//물건는 사과이다.
  X = apple ;
//그럼 부엌에는 어떤 것들이 있는가?
//사과와는 다른 물건 다시 호출하기
REDO: location(X, kitchen)
//부엌에는 브로콜리가 있다.
EXIT:(6) location(broccoli, kitchen)
//물건은 브로콜리이다.
  X = broccoli ;
//부엌에는 어떤 것들이 있는가?
//브로콜리와는 다른 물건 다시 호출하기
REDO: location(X, kitchen)
//부엌에는 크래커가 있다.
```

```
EXIT:(7) location(crackers, kitchen)
//물건은 크래커이다.
  X = crackers ;
//더이상 부엌에는 다른 물건이 없다.
FAIL - location(X, kitchen)
//no는 값을 찾을 수 없을 때 호출함.
  no
```



[그림4] prolog를 이용한 프로그래밍



[그림5] 프로그래밍처리 도식화단계

4. 연구 결과 분석

본 연구는 Keris에서 제공된 비판적 사고력에 관한 보고서를 토대로 하여 연역적 논리와 귀납적 논리를 중심으로 재구성하였다. 하지만, 다양한 논리영역에서도 극히 일부분만을 채택하였고 학생들의 논리적 사고와 프로그래밍 교육의 연계성에도 미흡한 점들이 많이 잔존하고 있다. 하지만, 아동들의 발달 단계라든지, 과업 성취력의 이해와 교수학습방법론에서의 체계적인 연구가 보다 활발히 이루어 진다면 어렵게만 생각되었던 프로그래밍 교육도 접근이 용이하다고 생각한다. 그래서 교수학습 과정안처럼 체계적인 위계를 통해 단계 단계를 밟아나간다면 충분히 체계적인 학습이 이루어질 수 있다고 생각한다.

고학년 학생들을 중심으로 하여 프롤로그 프로그램을 이용하여 다음과 같은 단계를 얻을 수 있었다.

첫째는 사실과 규칙에 대한 이해단계(언어적 접근)로서 학생들이 알아야할 규칙을 설정하는 일이다. 이 과정에서는 <약속하기>를 통해 아동들이 이해하고 알아야 할 사실에 대한 학습단계이다. 두 번째는 사실 관계를 정립하기 위한 도식화 단계(기호적 접근 : 논리구조 파악)인데 논리적 관계의 정립을 그림으로 나타냄으로서 보다 빠르게 논리성을 파악하는 단계이다. 세 번째는 간단한 문장을 기호화로의 변환 단계이다. 네 번째는 프로그래밍된 사실과 규칙에 대한 이해 단계이다. 다섯 번째는 프로그래밍된 서술구조에서 논리구조 파악단계이다. 다섯 번째는 프로그래밍을 통한 처리구조 도식화 단계이다.

이와 같은 단계는 문자로 구성된 프로그래밍의 마인드를 보다 시각적 효과를 이용하여 체계적인 접근이 용이하며 기호화를 통해 영어를 어려워하는 아동들에게 학습이 보다 쉽게 이루어지게 하였다. 하지만, 이 6단계의 과정은 프로그래밍 교육의 하나의 예일 뿐 검증된 바는 없고 단지 일반 교과에서 이루어지는 교수학습 과정 안을 프로그래밍교육에도 적용해 본 결과이다.

5. 결론

본 연구는 초등학생들의 논리적 사고력을 신장시키기 위해 지식 기반 프로그램인 선언적 프로그램을 통해 교육현장에서도 적용할 수 있는 프로그래밍 교육을 제언하고자 하였다. 학생들에게 논리적 사고를 분석적 방법을 통해 실제 프로그래밍을 해 봄으로써 연역적 사고 또는 귀납적 사고를 보다 효과적이고 체계적인 프로그래밍을 할 수 있도록 지도함으로써 제 8차 교육과정에서의 컴퓨터 교육과정의 일부분으로서의 프로그래밍의 마인드를 제시하였다. 초등학교 고학년 학생들 몇 명을 대상으로 교육해 본 결과 학생들의 논리적 사고를 발달시키기 위한 선언적 프로그래밍 교육에서 뿐만 아니라 다른 타 프로그래밍 교육에서도 다음과 같은 단계를 통해서 학생들을 지도하는 게 좋은 방법 중의 하나인 것 같다. 하지만 본 연구에서의 6단계의 과정은 프로그래밍 교육의 하나의 예일 뿐 검증된 바는 없다. 그러나, 실제 몇 몇 학생들에게 적용해 본 결과 다음과 같은 결론을 이끌어 낼 수 있는 데 생소한 프로그래밍을 통해 논리적인 사고를 이끌어 내기 위해서는 가장 기본적인 구조 파악 단계를 거쳐야 하는 데 바로 그림과 생각의 과정의 표출이 그 첫 번째이고, 이런 사실들의 단순한 열거가 아닌 짜임새 있는 재구성이 학생들의 논리적인 사고를 향상시킨다. 하지만 다분히 임의적이고 논리성의 영역도 변별성에 문제가 제기될 수 있다.

지금까지의 프로그래밍의 교육에서나 학문적 접근에서의 논리적 사고력 신장 교육은 정확한 아동의 고등 사고력을 이끌어 내지는 못했지만 이러한 연구 과정을 지속함으로써 보다 더 체계적인 컴퓨터 교육이 이루어질 수 있다고 생각한다.

참고문헌

[1] 이유순, "논리적 사고력 및 문제해결력 신장을 위한 컴퓨터 프로그래밍 교육", 이화여자대학교 대학원 석사논문, 1995

[2] 한국교육개발원, "사고력 검사연구(Ⅱ)", KERIS 연구보고 RRE 2002-3, 2002

[3] 이태욱, "컴퓨터 교육과 관련된 한미 교육 과정 비교분석에 관한 연구", 정보과학회지 p14(12),1996

[4] Bruner, J. Going Beyond the Information Given. New York: Norton. 1973.

[5] Papert, S. Teaching children to be mathematicians versus teaching about mathematicians*. International Journal of Mathematics Education, Science and Technology, 3, 249-262. (1972).

[6] Papert, S. Mindstorms: Children, Computers, and Powerful Ideas. NY: Basic Books. 1980.

[7] Csikszentmihalyi, M., and E. Rochberg -Halton. The Meaning of Things: Domestic Symbols and the Self. New York: Cambridge U. Press, 1981.

[8] 김수환, "논리적 사고력 신장을 위한 로고 프로그래밍 교육 활동의 효과의 분석". 한국교원대 대학원 석사논문, 1991

[9] Newell, A. and Simon, H.A. Human Problem Solving. New Jersey:Prentice Hall.(1972).