韓國數學教育學會誌 시리즈 E
<數學教育 論文集>
제 19집 제 2호, 2005. 7. 363-378

J. Korea Soc. Math. Ed. Ser. E:
Communications of Mathematical Education
Vol. 19, No. 2, Jul. 2005. 363-378

두리틀(Dolittle) 프로그래밍 활동을 통한 함수 개념 형성에 관한 사례 연구[1)]

조 영 주 (고려대학교 대학원)
김 경 미 (고려대학교 대학원)
황 우 형 (고려대학교)

본 연구는 객체지향형 교육용 프로그래밍 언어인 두리틀(Dolittle)을 수학교수-학습에 활용하기 위한 연구의 일부이다. 본 논문에서는 세 명의 고등학교 1학년 학생을 대상으로 7차 교육과정상의 중등 함수 단원을 중심으로 함수의 그래프에 대한 두리틀 프로그래밍 활동을 안내적 교수법으로 진행하고 그 결과를 분석하여, 두리틀 프로그래밍 활동이 함수의 개념 형성에 미치는 영향을 관찰하고 컴퓨터 친밀도와 수학적 성향이 프로그래밍 학습에 어떠한 영향을 주는지에 관하여 고찰하였다. 연구 결과, 두리틀을 이용한 함수의 그래프 그리기 활동은 학생들에게 함수의 기본 개념과 그래프의 성질을 이해하는데 효과적이었으며, 두리틀 프로그래밍 탐구 활동에 있어 학생들의 수학 성취도보다는 수학에 대한 긍정적인 성향과 컴퓨터와의 높은 친밀도가 긍정적인 영향을 미친다는 사실을 확인하였다.

Ⅰ. 서 론

수학 교수-학습에 있어 컴퓨터와 계산기 같은 테크놀로지의 활용은 현대 사회의 급속한 변화로 많은 사람들이 그 필요성을 인식하고 있다. 이제는 수학교육에서도 테크놀로지 활용에 대한 찬반 유무를 따지는 것보다는 어떻게 하면 테크놀로지를 수학 교수-학습에 잘 적용하여 교수-학습 효과를 극대화시킬 수 있는지에 대하여 연구해야 하는 단계에 와 있다고 하겠다. 7차 교육과정에서도 기술공학적 교구의 활용을 권장하고 있고, 정보화, 인터넷, 컴퓨터 프로그래밍 교육을 강조하고 있다. 이에, 수학과 컴퓨터 프로그래밍을 동시에 지도할 수 있는 수학 교수-학습 도구의 개발은 수학 및 정보교육에 있어서 매우 의미 있는 일이라 할 수 있다.

두리틀(Dolittle)은 다중 명령어 방식의 한글 프로그래밍 언어로 동일한 의미의 다중 명령어 집합에서 학습자가 자신의 인지 수준에 맞는 명령어를 선택하여 사용할 수 있고, 한글 어순의 간결한 문장으로 표현되기 때문에 초등학생들도 쉽게 프로그래밍 할 수 있는 언어이다. 또한 기존의 타 언어에 비하여 함수적 표현이 용이하며 거북 그래픽스(turtle graphics) 환경으로 그래픽 표현도 용이하다(김경미, 2004). 두리틀은 컴퓨터교육을 목적으로 개발된 언어이지만 많은 수학적 요소를 포함하고 있으며, 수학 교육의 활용 범위가 기존의 프로그래밍 언어에 비하여 매우 광범위하므로 수학교육적

1) 본 연구는 2004년 한국학술진흥재단의 지원(KRF-2004-030-B00051)으로 수행되었음,

활용에 긍정적인 효과가 기대된다. 본 연구는 고등학교 함수 단원을 중심으로 함수의 그래프에 대한 두리틀 프로그래밍 활동을 안내적 교수법으로 진행함으로써 학생 스스로 자신의 학습과정에 대해 인지적 모니터링을 할 수 있도록 하였다. 본 연구에서 다루게 될 연구 문제는 다음과 같다.

1. 두리틀 프로그래밍 활동이 학생들의 함수 개념 형성에 어떤 영향을 주는가?
2. 학생들의 컴퓨터 친밀도와 수학적 성향은 프로그래밍 학습에 어떤 영향을 주는가?

Ⅱ. 이론적 배경

1. 프로그래밍 활동과 수학 교육

컴퓨터 프로그래밍 학습 과정은 문제 상황 해결을 위해서 도전적이고 역동적인 학습 환경을 제공함으로써 학생들의 능동적 사고력을 촉진시킨다. 따라서 컴퓨터 프로그래밍 학습은 학습 방법의 여부에 따라 학생들의 문제 해결 능력을 발달시킬 수 있는 가능성을 가지고 있다(Papert, 1980; Clements, 1987, 1990; Seidman, 1987).

ACM(Association of Computing Machinery)에서는 K-12에서 제시한 교육과정에서 K-8(우리나라 중학교 2학년) 학생들에게 알고리즘적 사고 형성을 위해 LOGO와 같은 프로그래밍 언어 교육이 필요하다고 하였다(ACM, 2003). 컴퓨터과학 교육을 위해서 학생들에게 컴퓨터 프로그래밍을 가르치는 것은 수학 교육에 있어서도 긍정적인 요소를 가진다(Mibrandt, G., 1995; Vicki L. Almstrum 외, 2002; Kris Howell, 2003). 프로그래밍 과정은 학생들의 문제해결력과 알고리즘적 사고력을 향상시키며 특히 오류 수정 과정을 통해 반영적 사고를 증진시킨다(George Lukas 외, 1986; Papert, 1980; Ursula Wolz 외, 1994; Judith Gal-Ezer 외, 1999; 백영균 외, 1994). 그러나 우리나라에서는 수학 교수-학습에 프로그래밍을 적용하려는 시도가 매우 미흡하고, 수학교육에 활용된 프로그래밍 언어도 극소수이다. 따라서 수학 교수-학습에 활용 가능한 프로그래밍 언어의 발굴과 프로그래밍 활동에 관한 교수학적 연구가 꾸준히 이루어져야 한다. 특히, 수학 교수-학습을 위한 프로그래밍 활동에 있어 유념해야 할 것은 프로그래밍 활동이 특정한 컴퓨터 언어의 문법 학습보다는 프로그래밍 활동에 내재된 중요한 수학적 내용에 초점을 두어야 한다는 것이다.

2. 인지적 모니터링을 위한 안내적 교수법

최근 교육학자와 심리학자들은 역동적이고 도전적인 컴퓨터 프로그래밍 교수-학습 환경을 통해 인지적 모니터링 전략을 강화시키기 위해서 노력하고 있다. 인지적 모니터링(Cognitive Monitoring)이란 용어는 1980년대 중반부터 심리학계에 대두되었는데, 용어의 정의를 보면 학습자가 문제를 푸는 동안 진행되고 있는 자신의 인지적 과정(cognitive process)을 의식적으로 관찰, 통제하고 평가하

는 것으로 해석된다. 인지적 모니터링의 주요 요소로는 문제의 인식, 결과에 대한 예측, 문제 해결 방법의 모색, 계획의 수행, 그리고 나타난 결과에 대한 평가 및 수정 등을 들 수 있으며 교육자들과 인지 심리학자들은 인지적 모니터링을 고등정신 기능과 문제 해결 능력의 핵심 요소 중의 하나로 간주하고 있다. 더욱이 인지적 모니터링은 개인의 지능과 효율적인 사고력의 개인적 차이를 결정하는 요인 중의 하나이다(Brown, 1983: Sternberg, 1984, 1987).

안내적 교수법은 일반적으로 두 가지 요소가 조화를 이룬다. 첫째, 교수설계에서 뚜렷한 학습목표가 제시되며, 특정한 인지전략(cognitive strategy) 또는 메타인지 전략(meta-cognitive strategy)이 제시된다. 교수설계에는 또한 체계적이며 명백한 교수과정과 상황이 설계된다. 둘째, 이러한 교수설계와 과정이 실제 교실에서 교사의 중재로 학습자에게 전달되는 것으로 정의된다(Feuerstein, 1979; Missiuna 외, 1987; Samuels, 1986). 즉 안내적 교수법은 명백한 교수 설계와 중재적 학습이 결합한 개념이라 할 수 있다. Salomaon, Perkins(1987)와 Singley, Anderson(1989)에 의하면, 체계적으로 구성된 안내적 교수는 학생들이 배워야 할 복잡한 지식과 정보를 관리, 통제하는 데 도움을 준다. 즉, 특정한 학습전략의 모델이 명백하게 교수 설계 과정에 제시되므로 학습자가 주어진 내용을 학습해 가는데 전략을 활용할 수 있게 한다. 또한 이렇게 설계된 교수설계의 모델은 교사의 중재적 학습 활동을 통하여 학습자에게 전달된다.

중재적 학습은 교사가 일방적으로 학생들에게 지식이나 정보를 전달하고 기억하게 하는 교사 중심의 학습을 배제하며, 또한 학습자 스스로 자기 발견적 학습에만 의존하여 학습하는 학생만의 학습도 배제한다. 중재적 학습에서 교사와 학생은 모두 중요한 의미를 가진다. 교사는 새로운 학습전략, 정보를 전달하는 데 있어 소크라테스의 대화법(Socratic dialogue)을 사용하는데 이는 학생들 스스로가 의식적으로 주어진 문제 상황에 관심을 갖도록 유도하기 위한 것이다. 이것은 학생들이 학습의 방향을 찾지 못하고 방황할 때 유도적 질문으로 학생들이 가야 할 방향을 제시하는 것이다(조미옥, 1991).

이러한 안내적 교수법은 학생들이 의식적으로 지식, 정보 또는 학습 전략을 습득해 나가도록 도울 뿐 아니라, 학생들에게 습득된 지식 및 전략을 다른 새로운 문제 상황에 전이할 수 있도록 해준다.(Salomon 외, 1987; Swan 외, 1989). 따라서 안내적 교수법을 사용한 두리틀 수업에서 교사는 언어의 문법적인 요소만을 제공하고, 문제 해결 방법을 제시하지 않았으며, 학생 스스로 문제를 탐구하고 여러 번의 시행착오를 통해 프로그래밍을 하도록 유도하는 것에 초점을 두었다. 특히 교사는 인지적 모니터링 전략을 발달시키기 위해서 인지 모니터링 활동 단계의 특징들을 이해하여 전략적으로 제시하고, 소크라테스적인 대화법을 사용하여 학습의 중재자로서의 역할을 수행하고자 하였다.

<표 1> 인지 모니터링의 활동 내용

인지모니터링 과정	설명	유도 질문
문제 분석	주어진 문제의 필수요소들과 단서들을 정의하고 문제해결에 핵심이 되는 요인들을 분류, 파악한다.	"이 문제를 어떻게 해결할까?" "이 문제에 대해 우리가 알고 있는 것은 무엇일까?" "이 문제에서 중요한 요소들은 무엇일까?"
계획	전 단계에서 분석되어지고 나누어진 정보, 요소 및 단서 등을 조직하고 순서적으로 구성하여 그 해결점을 찾는다.	"이 문제를 풀기 위해 분리되어진 각 요소들을 어떻게 연관지을 수 있을까?" "지금까지 모아진 정보들을 좀 더 효율적으로 구성할 수 있는 방법은 무엇일까?" "이 문제를 해결하기 위해 어떤 전략이 필요할까?"
실행	학습자가 세운 계획을 명령어 입력을 통해 실행한다.	"거북이에게 우리 계획을 이해하고 실행하도록 하려면 두리틀 명령어로 어떻게 기술하면 될까?"
오류검정	첫째, 문제해결 계획에 의하여 산출된 결과와 원래 달성해야 할 목표를 비교하여 차이점을 설명한다. 둘째, 차이점에 준거하여 프로그래밍 과정의 오류가 어디에서 발생되었는지 추측하여 지적한다. 셋째, 지적된 프로그래밍의 절차에서 무엇이 잘못되었는가를 설명한다.	"무슨 일이 생겼니?" "의도했던 결과와 실제 결과의 차이는 무엇이니?" "어디에서 오류가 생겼다고 생각하니?" "계획대로 잘 진행되고 있니?" "분리한 요소들은 적절하니?" "무엇이 잘못되었는지 설명할 수 있니?"
오류수정	지적된 문제 과정을 수정하고 그 결과를 관찰한다.	"어떻게 바꾸었니?" "적절하게 수정되었니?" "다른 방법은 없을까?"

3. 객체지향형 교육용 프로그래밍 언어 두리틀(Dolittle)

두리틀은 LOGO의 거북 그래픽스(Turtle Graphics)와 인크리멘탈(Incremental) 프로그래밍 방식, 즉각적인 피드백 등 많은 교육적 이점을 수용하고, 현대 프로그래밍의 고급 기능들을 프로토타입(Prototype) 방식을 통해 어린 학생들도 쉽게 이해할 수 있게 한 텍스트 기반의 한글 교육용 프로그래밍 언어이다. 종래 교육현장에서 사용되어 왔던 Basic과 LOGO가 플로우차트로 대표되는 '리스트(List)'라는 '계산기의 원리'를 중심으로 프로그램을 기술하는 것과는 다르게 두리틀은 동물과 버튼 등 실세계의 '사물'에 해당하는 '객체(Object)'를 단위로 한 프로그램을 기술하는데 이것은 비교적 인간과 가까운 사고방식으로 프로그램을 한다는 점에서 현대 컴퓨터 언어에서 지향하는 패러다임에 부합된다(가네무네, 2003). LOGO에서는 거북 객체 하나만을 사용하여 도형을 그리는 반면, 두리틀은

여러 거북 객체를 생성하여 각 객체에게 역할을 분담하여 그리거나 혹은 어떤 기능을 수행하게 할 수 있으며, 거북 객체 뿐 아니라 버튼, 텍스트 필드, 타이머, 리스트, 배열, 라벨, 선택 메뉴, 시리얼 포트, 멜로디 등 다양한 객체를 통해 기능적으로나 시각적으로 다양한 프로그램 작성이 가능하다(김경미, 2004; 황우형 외, 2004).

특히 두리틀은 수학의 함수영역 교수-학습에 긍정적인 효과를 얻을 수 있는 요소들이 많이 잠재되어 있다. 프로그래밍 과정에서 사용되는 메소드의 개념은 함수의 개념과 거의 유사하기 때문에 여러 메소드를 만들어 사용하는 동안 학생들은 자연스럽게 함수의 개념을 습득할 수 있고, 변수들의 조작을 통해 변수의 동적변화성을 이해할 수 있다. 또한 두리틀에서는 학교 수학시간에 사용하는 친숙한 한글 수학용어를 변수로 사용할 수 있으며, 수식 표현이 수학에서의 수식 표현과 거의 유사하므로 학습 연계의 효과를 얻을 수 있다. 그리고 다양한 객체와 메소드를 사용하여 쉽게 학생 스스로도 자신의 방법을 개발하여 프로그래밍할 수 있기 때문에 문제 해결력 및 창의력 개발에 효과가 있을 것이라 기대된다.

Ⅲ. 연구 방법

본 연구는 교사가 개별 학생의 교수-학습 과정에서 중재자의 역할을 수행하였고; 학생의 행동을 관찰, 서술, 해석하는 질적 사례 연구(Qualitative Case Study)로 진행되었다. 참여관찰과 인터뷰를 통하여 자료를 수집하고 해석적 과정을 통하여 자료를 분석하였다. 참여관찰과 면담을 병행한 이유는 언어적 표현과 비언어적 표현의 불일치로 인하여 사고와 행동이 다르게 나타난 현상을 규명하고 사실을 보다 정확히 파악하기 위해서이다.

1. 연구참여자

수도권 고등학교 1학년 학생들 중에서 1학기 중간·기말고사의 수학 시험 결과에 기준하여 상·중·하에 속하는 학생 3명을 의도적으로 결정하였다. 연구 참여자에게 두리틀에 관한 내용과 연구가 언제, 어떻게 이루어질 것인지, 연구를 통해 얻을 수 있는 이점이 무엇인지에 대하여 설명한 후 연구 참여를 허락 받았고, 각 참여자의 학부모에게 동의를 얻은 후 연구를 시작하게 되었다. 연구 참여자 3명의 학생은 S1, S2, S3(연구 참여자의 보호를 위해 모든 학생의 이름은 가명을 사용하였다.)이며, 이들의 학업 성취는 수학교과에서 상-중-하의 수준을 나타내었다. 학습은 개별학습의 형태로 이루어졌으며, 학생들 간의 토론 시간을 따로 주어 학생들 간의 상호작용을 관찰하였다. 첫 번째 면담을 통해 알게 된 학생 참여자들의 개별적인 특성은 다음과 같다.

<표 2> 함수 영역 연구 참여자의 배경

성명(가명)	성별	학년	컴퓨터 사용 시간(1주)	프로그래밍 활동 여부	수학적 성향	수학 성적 평균(3회)
S1	남	고 1	10시간 정도	×	긍정적	100점(상)
S2	남	고 1	30분 미만	×	부정적	70점(중)
S3	남	고 1	3시간 정도	×	긍정적	58점(하)

S1은 학업성취도가 "상"인 학생이지만, 평소 컴퓨터 오락을 즐기며, 바빠도 하루에 1-2시간은 꼭 게임을 하는 학생이다. 수학뿐만 아니라 모든 과목에서 학업 성취가 "상"에 속하는 학생으로, 수학 과목에 대한 긍정적인 성향을 나타냈으며, 함수 부분 전 영역에 대한 선행학습이 완전히 되어있는 학생이었다.

S2는 학급의 반장으로 활동하는 학생이지만 성격이 외향적이라기보다는 내성적인 면이 강한 학생이다. 컴퓨터와의 친밀성도 세 명의 학생 중 가장 낮아서 집에서는 거의 컴퓨터를 하지 않으며 공부를 강조하는 학교생활에 상당히 피로함을 느끼고 있는 학생이었다. 수학과목에 대한 성향도 부정적이고, 학업성취가 "중"으로 학원에서 선행학습을 하였으나 수학적 개념이 다소 부족한 학생이었다.

이 학생과 비교해서 S3은 수학 학업성취가 "하"인 학생임에도 불구하고, 수학에 대해서는 긍정적인 성향을 가지고 있는 학생이다. 컴퓨터 오락을 매우 즐기는 편이고, 실수나 문제 풀이의 오류에도 별로 개의치 않고 자기가 생각한 바를 주저함 없이 말하고 질문하였다. S3은 함수에 대한 선행학습이 이루어지지 않았을 뿐 아니라 중학교 함수 개념조차도 형성되어 있지 않은 학생이었다.

본 연구에 참여한 학생들을 학업성취에 따라 상, 중, 하로 구분하여 선정한 것은 두리틀 프로그래밍 활동이 학생의 학업성취에 관계없이 모든 학생들이 쉽게 습득할 수 있는 언어인지를 알아보기 위한 것으로 각 수준의 학생들의 반응을 알아보고자 하는 연구의 의도를 반영한 것이다.

2. 연구절차

2004학년도 제 2학기인 9월 1일부터 12월 30일까지, 매 차시가 80분 정도 소요되는 8차시로 이루어진 변수 및 함수에 관한 수업을 실시하고, 관찰 8회 반구조화면담 3회를 실시하였다. 두리틀 활동은 정규 수업 중에는 수행할 수 없었기 때문에 방과 후 보충수업이나 공휴일을 이용한 보충수업의 형태로 실시하였다. 연구 전반부의 1~3차시는 두리틀 프로그래밍의 기초를 익히는 강의식-실험 수업으로 진행하였으며, 4~8차시는 연구 참여자가 직접 프로그래밍을 창작하는 안내적 교수법으로 진행하였다. 수업이 진행된 구체적인 일정과 수업 내용은 다음과 같다.

<표 3> 함수 적용 연구 일정

일정	차시별 주제	일정	차시별 주제
Orientation	동의서 작성 학생 면담	5차시	이차함수의 그래프 그리기
Pretest	실험 전 변수 및 함수의 개념에 관한 사전평가 실시	6차시	이차함수의 평행이동, (S1: 3차, 4차 함수, 유리, 무리함수의 그래프 그리기)
1차시	두리틀 프로그래밍 언어의 구조, 특징, 기본도형 그리기	7교시	삼각함수의 그래프 그리기
2차시	원 그리기 · 도형 옮기기	8차시	삼각함수 그래프를 이용하여 최대, 최소, 주기 구하기
3차시	메소드(Method) 정의하기		학생 면담
4차시	일차함수의 그래프 그리기 학생면담	Posttest	교수실험을 통한 변수 및 함수 개념의 이해 정도를 평가 실시

8차시 수업 대부분은 비디오카메라에 의해 녹화되었으며, 수업이 끝난 후 관찰일지에 코딩을 하여 자료를 분석하고 다음 차시 수업을 준비하였다. 관찰 시 알기 힘든 부분이나 중요한 분석요소는 분석틀에 기록하였다가 학생 개인 면담 시간을 활용하여 다시 질문하였고 매우 유용한 정보를 얻을 수 있었다. 자료 분석은 수업 이후 작성한 관찰 일지와 매 차시 학생들이 작성한 프로그래밍과 활동지 그리고 면담내용을 통해 범주화되고 분석되었다.

Ⅳ. 연구결과 및 분석

1. 메소드 정의와 변수 개념

3차시 메소드 정의 학습 이후 4차시부터 본격적인 함수의 그래프 그리기 활동을 시작하였으며, 메소드 개념까지만 알려준 상황에서 학생 스스로 함수의 그래프를 그릴 수 있을지를 알아보기 위하여 함수의 그래프 그리기 소스를 공개하지 않았다. 상, 중, 하 학생 S1, S2, S3은 4차시 수업에서 일차함수 y=x 그래프 그리기 활동을 하면서 매우 막막해 했으며, 오랜 시간 동안 고민하였고, 진전이 없었다. 그러나 여러 번의 시행착오를 겪은 후에 함수에서의 변수와 두리틀 프로그래밍의 메소드의 개념을 연결함으로써 S1과 S3는 함수의 그래프를 혼자 힘으로 그리는 데 성공하였다. 학업 성취 "상"인 S1은 예상대로 일차함수 개념을 메소드 개념으로 바꾸어 그래프를 그렸으며, 함수 개념이 제대로 형성되어 있지 않은 S3은 연구자가 알기 힘든 독특한 방식으로 그래프를 그렸다.

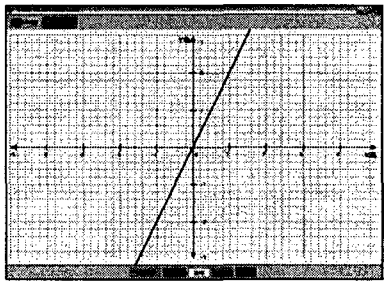
교사 : y=2x의 그래프를 그리는 문제를 어떻게 해결할까?
S1 : ……

교사 : 이 문제에서 중요한 요소들은 무엇일까?
S1 : 기울기가 2이고 … x가 1만큼 증가하면 y는 2만큼 증가해요.
교사 : 가장 적절한 거북이의 시작점은 어디일까?
S1 : 함수 정의역의 가장 작은 값이 아닐까요. 계속 1만큼 증가시키면 되니깐요.(편집창에 작성한 후 실행창을 열어 확인한다.)

편집창

```
좌표=거북! 만들고 "좌표_1.gif" 변신.
두리=거북! 만들다.
일차함수 = [|x|
            [y=2x.
            ! (x) (y) 위치한다.
            x=x+1.]! 1000번 반복한다.
            ].
두리! -500 일차함수.
```

실행창



<그림 1> S1이 작성한 일차함수 y=2x 그래프

S1은 면담에서 자신의 프로그래밍에서 -500이 의미하는 것이 함수의 정의역의 왼쪽 값이라는 것을 알고 있었으며, 대칭된 함수를 그리기 위해서 1000번 반복하였다고 하였다. 그리고 두리틀 화면이 직교좌표이므로 "(x) (y) 위치하다"를 이용하여 거북이의 위치를 지정해주었다고 하였다. S1은 면담에서 일차함수 메소드 정의에서 인수(Parameter) x에 따라 y값이 함수식에 의해 정의되는 것이 함수의 개념과 유사하다고 말하였다.

관찰과 면담 및 학생의 활동지를 분석한 결과 S3은 함수의 대응관계와 정비례 관계를 알지 못했으며, 순서쌍으로 함수를 그리는 것을 두리틀 프로그래밍에 적용하지 못하였다. 그 대신에 "이동하다"는 명령어를 통해 거북이의 위치를 평행이동 하였으며, 직관에 따라 거북이의 시작점을 계산하여 넣어주었다. 다음은 유사한 과정을 겪은 S3의 y=x의 프로그래밍 결과와 대화내용이다.

S3는 처음 y=x 함수를 프로그래밍할 때, x=y로 기술하여 그래프를 그리지 못하였다. 교사의 지도를 통해 y=x로 변경하여 그래프를 그렸는데, 처음에는 자신이 작성한 프로그래밍이 어떻게 실행되는지도 이해하지 못하였다. 그래서 연구자는 x, y 관계표를 그리게 하여 기술된 프로그래밍의 과정을 통해 얻어지는 x, y값을 활동지에 쓰도록 하였다.

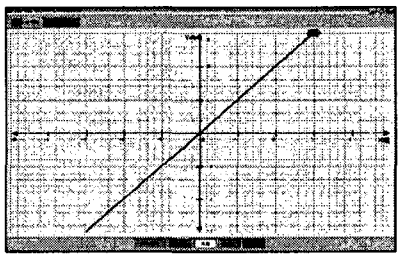
교사 : 너가 어떻게 프로그래밍 했는지 설명해줄래?
S3 : 여기요, 좌표평면을 그리고요, 가운데가 (0,0)이고요, 처음 거북이는 (한 점을 가리키며) (-300,-300)으로 가요. 그리고 일차함수라는 메소드를 만들어서 x하고 y에 함수값을 줘요.
교사 : 함수가 뭐라고 생각하니?
S3 : 어어, 함수가요? y값 아닌가요? 함수가요? … 잘 모르겠는데요.

교사 : 그럼, 너가 그리려고 하는 y=x라는 함수가 뭔지 모르니?
S3 : 네 (쑥스러운 표정을 지으며)
.....(중략).....
교사 : 자, 그럼 y=2x라는 그래프를 그려볼 수 있겠니?
S3 : 예, 이 값만 바꾸면 되지 않나요? 어? 이상한데요.(맨 마지막 줄 [두리! 1 1 일차함수.]를 [두리! 1 2 일차함수.]로 바꾼다. 실행 결과가 원하던 그래프로 나오지 않자 당황해하며 [두리! 2 1 일차함수.]로 바꿔본다.)

편집창

```
좌표=거북! 만들고 "좌표_1.gif" 변신.
두리=거북! 만들다.
두리! 펜들기 -300 -300 위치하고 펜내리기.
일차함수 = [|x y|
        [y=x.
        ! (x) (y) 이동한다.]! 600번 반복한다.].
두리! 1 1 일차함수.
```

실행창



<그림 2> S3이 작성한 일차함수 y=x 그래프

S3는 처음 "일차함수"라는 메소드를 통해 인수 x를 받아 y의 값을 x에 관한 식으로 정의해주는 것을 이해하지 못하였으며, 함수의 개념이 형성되어 있지 않았다.

컴퓨터 프로그래밍에 있어 메소드 개념은 수학에서의 함수의 개념과 유사하다. 그러므로 함수의 개념이 형성되어 있는 학생과 그렇지 않은 학생은 메소드 정의에 있어 큰 차이를 나타내었다. 함수에 있어 변수 개념의 형성 또한 매우 중요한 부분이다. 김남희(1997)는 변수의 동적 변화성을 가장 자연스럽게 다룰 수 있는 단원으로 함수단원을 언급하고 있다. 학교수학에서 변수에 대한 동적인 접근은 함수에서 즉, 두 가지 변수가 서로 관련을 맺고 있을 때 한 변수를 다른 변수와 비교하여 그것의 변화성과 공변성을 강조하는 경우에 두드러지게 강조될 수 있다. 실제로 학교수학에서 변수에 대한 형식적 정의를 함수 단원에서 제시하고 있다. 본 연구에서 학생들은 메소드의 정의에 관한 수업 이후 변수란 많은 수를 대체할 수 있는 것이고, 어떤 문자를 사용하여도 상관이 없다는 변수의 자유성을 인식하였다.

2. 함수의 개념 형성

두리틀의 함수 영역 적용 결과 두리틀을 이용한 함수의 그래프 그리기 프로그래밍 활동은 학생들에게 함수 정의, 정의역, 치역, 정비례와 반비례 관계와 같은 함수의 기본 형성에 긍정적인 영향을 주었다. 다음은 학업 성취가 낮은 S3 학생이 두리틀 프로그래밍을 통해 함수의 그래프를 그리는 활동에서 여러 번의 시행착오 결과 스스로 함수의 개념을 형성한 부분이다. 다음은 일차함수 y=4x 의

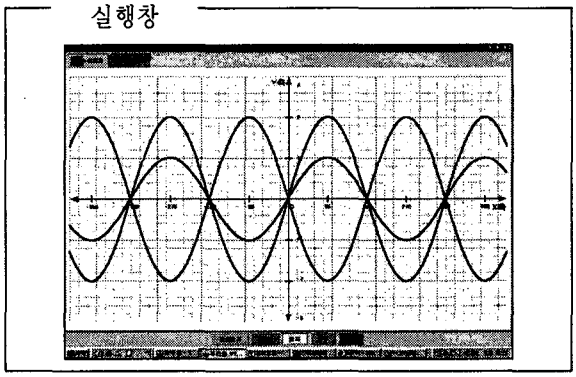
그래프를 프로그래밍하는 과정 중에 학생 S3이 자신의 프로그래밍을 인지적 모니터링하고 있는 장면이다.

교사 : 어떤 규칙을 찾았니?
S3 : y=4x 할 때요 -100 -400 할 때 y값이 100만큼 줄어드는 게 아닌가 라는 생각을 했거든요. y=5x 만들 때요. -100 -500 으로 했더니 딱 되는 거예요.. 그래서 와~ 이거 y값을 100만큼씩 여기서부터 줄여도 되는구나. 처음만 조심하면 되겠구나 라고 생각했거든요. 근데 오늘 와서 생각해 보니깐요.
처음 위치하는 값이요. 함수 y=x 일 때, -300 -300 은 일 대 일(1:1), 그리고 y=2x 일 때, -200 -400 은 일 대 이(1:2), y=3x 일 때, -100 -300에서 일 대 삼(1:3), 이런 식으로 비례가 되는 거예요. 아.. 그래 가지고 아 이거는 증가한다는 그런 법칙이 아니라 함수가 그 함수 값에 따라 비례한다고 생각했죠. 그런 식으로 생각했어요.

S3은 일차함수의 기울기와 y절편의 개념조차 없는 상태에서 일차함수가 비례관계라는 것을 프로그래밍의 시행착오를 통해 스스로 터득하였다. 그 이후에도 S3은 다양한 함수의 그래프를 그리는 프로그래밍 과정을 통해 초기 "이동하다" 명령어를 사용한 프로그래밍 방식에서 함수의 개념이 내포된 프로그래밍 방식으로 변하였으며, 마지막 면담에서 자신의 학업 성적에 이번 두리틀 탐구 수업이 매우 큰 영향을 주었다고 하였다. 본 연구의 두리틀 프로그래밍 활동이 S3의 함수 개념 형성에 긍정적인 영향을 미쳤다고 하겠다. S3 뿐만 아니라, 초기에 부진한 결과를 보였던 S2도 두리틀 탐구 수업을 통해 삼각함수의 개념을 학습하는 좋은 결과를 보였다. 다음은 S2가 교사와 y=sin(x), y=2sin(x), y=-2sin(x) 의 그래프를 프로그래밍하면서 나누었던 안내적 교수의 장면이다.

교사 : y=sin(x)와 y=2sin(x)를 같이 그려보자. (S2가 그리는 것을 지켜본 후) 지금 이 두 그래프를 함께 보고 있는데, 차이점이 뭐라고 생각하니?
S2 : 최대값, 최소값이 달라요. 1과 -1 그리고 2와 -2예요.
교사 : π/2 일 때의 함수값을 보면 y=sin(x)에서 함수값이 1이었는데, y=2sin(x)에서는 2배가 되어서 2가 되었지? 그러면 만약 y=2sin(x)가 아니고 y=a sin(x)라면 최대값은 얼마가 될까?
S2 : (머뭇거리면서) a.?
교사 : a이고 최소값은?
S2 : -a.
교사 : a=-2인 경우 y=-2sin(x) 그래프의 최대값은 얼마지?
S2 : (자신없어 하며) -2가 아닌데…. a가 아니라는 건데….
교사 : 편집창으로 한번 들어가서 확인해 보자.
· · · 중략 · · ·
교사 : 자 여기서 이것이 y=sin(x)일 것이고, y=-2sin(x)는 어느 것이지?
S2 : 손으로 짚는다.
교사 : 이럴 때 y=2sin(x)와 y=-2sin(x)의 그래프를 잘 봐. 의도했던 그래프와 실제 결과의 차이는 무엇이라고 생각하니?
S2 : 아! 절대값이예요. 최대값은 |a|이예요. 그래프를 그려보니까 확실히 알겠어요(웃는다).

교사 : y=2sin(x)의 주기를 이야기 하지 않았네. 주기가 얼마니?
S2 : 2π요. 주기는 y=sin(x)랑 같아요.



<그림 3> S2의 삼각함수 프로그래밍 결과

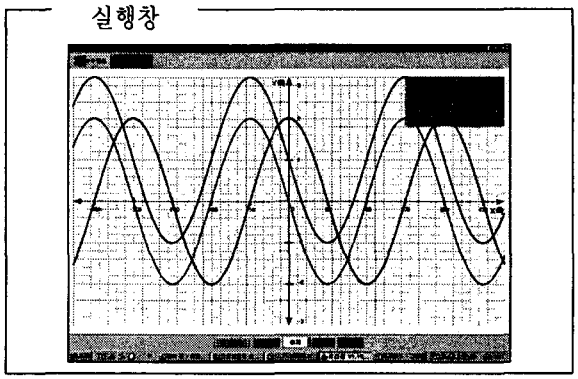
세 개의 그래프를 하나의 창에 프로그래밍하는 활동을 통해 S2는 전체적인 프로그래밍의 기법을 한 번에 터득하였고, 삼각함수의 그래프 프로그래밍뿐만 아니라, 삼각함수의 그래프에 대한 성질 즉, 정의역과 치역, 대칭점, 최대값과 최소값, 주기 등을 시각화를 통해 추론함으로서 귀납적인 방법으로 일반화하였다. 이 수업에서 S2는 본인이 어렵다고 생각했던 삼각함수의 그래프를 그렸다는 것을 몹시 신기해했으며, 안내식 발견수업을 통해 삼각함수의 원리와 개념을 스스로 터득해 나갔다.

다음은 S1 학생이 삼각함수의 그래프와 개념에 관한 파악이 끝난 후 친구들을 위한 삼각함수의 그래프 프리젠테이션 작품을 교사의 안내를 받으며 본인이 직접 만든 것이다.

교사 : 편집창에서 삼각함수의 식을 쓰면 실행창에 보여지게 되는데 네가 이 작업을 하면서 불편하다고 느낀 점이 있었니?
S1 : 실행창을 우리가 보는데요, 식이 같이 보였으면 좋겠어요. 선생님이 칠판에 쓸 때는 같이 쓰시잖아요.
교사 : 무엇을 이용하여 그래프와 함께 식을 적을 수 있을까?
S1 : 버튼을 이용하면 될 것 같아요. 버튼의 이름을 그래프의 식으로 주면 될까요?
교사 : 한번 해보자.
S1 : 버튼이 예전에 한 번 배운 거라 명령어가 좀 헷갈려요.
(S1은 「 버튼1 = 버튼! "y=2cos(x)" 만들다 270 200 이동 230 40 크기.
버튼1 : 클릭 = [냥! -500 코사인1]. 」라고 편집창에 썼다.)
교사 : 네가 지금 y=2cos(x), y=2cos(x+90°), y =2cos(x+90°)+1) 의 그래프를 하나의 실행창에서 보여주고 있는데 세 그래프의 구분이 잘 가지 않는 것 같구나. 어떻게 하면 되겠니?
S1 : 그래프의 색깔을 달리하면 될 것 같아요.
(말없이 편집창에 「양 = 거북! 만들다 (빨강) 색칠하다.
뇽 = 거북! 만들다 (파랑) 색칠하다.」를 쓴 후 실행 버튼을 눌렀다.)

교사 : 와! 놀랍구나. 이렇게 하니까 모든 그래프가 한눈에 금방 들어오는구나. 또 다른 방법은 없을까?

S1 : (칭찬에 좋아하며..) 생각해볼게요.



<그림 4> S1의 버튼을 이용한 cos 함수의 비교

S1은 삼각함수의 평행이동의 개념을 완벽하게 구현했을 뿐 아니라, 그래프에 색상을 달리하여 그래프를 구분하기 쉽게 하였다. 단순히 그래프 그리기의 수준을 넘어, 버튼 객체를 이용하여 원하는 삼각함수를 하나만 그릴 수도 있고 비교도 가능한 프로그래밍 작품을 완성하였다. S1은 이 작품을 완성한 후 자신이 하나의 프로그래밍 작품을 만들었다는 것이 매우 뿌듯하며 재미있었다고 하였다.

3. 수학적 성향과 컴퓨터 친밀도

함수 프로그래밍 과정에서 학생들 간의 개인차가 매우 크게 나타났다. 연구자가 예상했던 것과는 다르게 학업 성취도가 중위권인 학생(S2)보다 하위권인 학생(S3)에게서 학습 속도와 연구 참여의 적극성이 높게 나타났다. 상위권 학생(S1)은 예상대로 모든 수업을 잘 수행하였다. 연구자는 수학 학업 성취도와 프로그래밍 학습 사이의 관계를 알아보기 위해서 학생들의 기본 배경에 대한 자료를 분석하였고, 개인 면담을 실시하였다. 그 분석 결과 두리틀 프로그래밍 활동에 적극적이며 학습 결과가 좋은 S1과 S3은 수학 과목에 대한 긍정적인 성향을 가지고 있는 반면, 학업 성취가 중위권임에도 불구하고 탐구 수업에 소극적인 태도를 보였던 S2는 수학에 대하여 부정적인 성향을 가지고 있었다. 다음은 교사가 학생 세 명 모두에게 개별적으로 수학과목에 대하여 어떻게 생각하느냐는 질문을 했을 때 학생들의 답변이다.

S1(상) : 가장 깔끔한 과목인 것 같아요. 다른 과목은 공부를 해도 뭐가 뭔지 잘 모르겠는데 수학은 공부하고 나면 뭐였다는 것이 잘 보여요.

S2(중) : 사회에 나가서 수학이 그다지 사용되지 않고 필요도 없는데, 왜 이렇게 어려운 수학을 배우는지 이해가 안 가요. 수학을 못해도 사는 데 지장이 없잖아요.

S3(하) : 수학이라는 과목이요. 신비한 것 같아요. 깊이 있게 들어가면 갈수록 파헤쳐지는 게요. 점점 줄어드는 것이 아니라 많아진다고 할까. 사각형만 해도요. 안으로 깊숙이 들어가다 보면요. 법칙들이 많고 그런 식이 맘에 들어요. 수학은 그런 것 같아요.

S2는 수학이란 과목은 실생활에 필요하지 않은 과목으로 대학을 가기 위해서만 필요한 과목으로 여기고 있었으며, 학교를 졸업하면 수학을 공부하고 싶지 않을 정도로 수학에 대하여 부정적인 생각을 가지고 있었다. 수학에 대한 성향은 학생들의 학습 동기에 매우 큰 영향을 미치며, 학습 의욕과 매우 밀접한 관계가 있다. 이것이 프로그래밍 교수-학습에도 영향을 미친 것이라 생각된다. 학생들의 수학적 성향뿐 아니라 컴퓨터와의 친밀도가 학업 성취도보다 두리틀 프로그래밍 활동에 큰 영향을 주었다. 다음은 교사가 하루에 컴퓨터 사용시간을 묻는 질문에 대한 학생들의 대답이다.

S1 : 바빠도 하루에 1-2시간씩은 꼭 컴퓨터를 해요. 학교에서 저녁 교실 개방에 참여하구요, 학원 갔다가 집에 가면 12시가 좀 넘거든요, 그때부터 오락을 하는 거죠. (웃으면서) 그래도 그거 하면서 밤을 새진 않아요.

S2 : 교실 개방 끝나고, 밤 9시 이후에 다시 학원으로 가서 12시가 넘어서 집에 오는 데 오면 자기 바쁘죠. 저는 오락은 거의 안 해요.

S3 : 옛날에는 많이 했는데요. 요새는 고등학생이다 보니깐 일주일에 3시간 정도요. 하하. 이거 의외로 적은 거예요.

학생들의 수학적 성향과 컴퓨터 친밀도에 따라 초기 학습의 적극성과 학습 결과에 차이가 있긴 하였지만, 시간이 지날수록 그 차이는 줄어들었다. 학업 성취가 우수하고 수학 과목에 대한 긍정적인 사고를 가지고 있으며, 컴퓨터 게임을 좋아했던 S1의 경우는 다른 학생들보다 항상 뛰어난 학습 효과를 보여주었으며, 마지막 차시에 지금까지 배웠던 내용으로 삼각함수의 그래프를 버튼 객체를 사용하여 그려주는 프로그래밍을 만들어 보여주기도 하였다. 수학적 성향이 부정적이고 컴퓨터를 좋아하지 않았던 S2의 경우 연구 초반에는 다른 학생들보다 약간 뒤떨어졌지만, 동료 친구들과 교사의 도움으로 함수 프로그래밍의 개념을 인지한 이후부터는 S3보다 더 뛰어난 학습 효과를 보여주었다. 함수의 개념이 형성되지 않았던 S3 학생의 경우 자신만의 독특한 방식으로 함수의 개념을 사용하지 않고 함수의 그래프를 프로그래밍 하였는데, 여러 번의 시행착오를 겪으며 함수의 개념을 알게 되었고, 함수 개념이 내포된 프로그래밍 방식으로 프로그래밍 하는 것을 볼 수 있었다. 그러나 연구 후반으로 갈수록 학업 성취가 낮은 S3가 부진하였는데, 이는 연구 초반보다는 후반에 수학적 개념이 많이 포함되었기 때문으로 분석되었다.

V. 요약 및 결론

중등 함수 영역을 중심으로 두리틀 프로그래밍 활동의 수학교육 적용 가능성을 탐구한 본 연구는 연구를 진행하는 연구자나 학생들에게 새롭고 의미 있는 시간이었다. 두리틀 프로그래밍 탐구 활동

을 적용한 결과 다음과 같은 결론을 얻었다.

첫째, 객체지향형 교육용 프로그래밍 언어(Educational Programming Language:EPL) 두리틀은 수학 교수-학습에 적합한 언어이다. 연구가 진행되는 동안 두리틀 프로그래밍으로 함수의 표현이 용이함이 확인되었고, 학생들은 쉽게 언어를 습득하고 자신의 생각을 기술하며 탐구하는 과정을 보여주었다. 학생들은 내장된 수학 함수와 인지하기 쉬운 언어 표현으로 두리틀 언어 문법보다는 수학적 내용에 더욱 집중하였다.

둘째, 학생들의 긍정적인 수학적 성향과 컴퓨터와의 높은 친밀도는 두리틀 프로그래밍 탐구 활동에 긍정적인 영향을 주었다. 그리고 학생들의 성취도 차이는 시간이 흐르면서 점차 좁혀지는 것을 알 수 있었다. 수학적 성향이 긍정적이고 컴퓨터와의 친밀도가 높은 학생은 프로그래밍을 두려워하지 않았으며, 수학적 성향이 부정적이고 컴퓨터와 친밀도가 낮은 학생은 프로그래밍하는 동안 명령어 기술에 주저함이 많았다. 그러나 이러한 학생도 동료 학생과 교사와의 상호작용으로 극복이 가능하였으며 두리틀 프로그래밍을 하는데 낙오되지 않았다.

셋째, 두리틀 프로그래밍을 이용한 함수의 그래프 그리기 활동은 학생들에게 함수의 정의, 절대좌표의 인식, 정의역, 치역, 비례관계와 같은 함수의 기본 개념과 그래프의 성질을 쉽게 이해하게 하여 그래프 그리기 능력에 긍정적인 영향을 주었다. S3은 함수의 비례관계를 스스로 터득하였으며, 모든 학생들이 주어진 그래프를 그리는 프로그래밍을 완성하였고, 그 프로그래밍에서 함수의 정의역과 치역, 함수의 개념을 터득하였다. 특히 안내식 발견수업은 학생들이 시행착오를 거쳐 스스로 학습의 목표에 도달하는 교수-학습 방법으로 프로그래밍 활동에 적합한 것으로 나타났다.

학생들에게 수학적인 개념 지도를 함에 있어서 중요한 점은 정의에 의한 형식적인 지도뿐만 아니라, 직관이나 관찰을 통해 구체적인 상황에서 인식시켜야 한다는 것이다. 본 연구는 두리틀 프로그래밍 활동을 통해 복잡한 계산문제에 치우치지 않고, 원리와 개념 이해에 보다 많은 학습시간을 사용할 수 있었다. 두리틀 프로그래밍 활동은 학생들에게 하나의 구체적 조작물로 활용될 수 있는데, 두리틀 환경은 학생들이 그들의 비형식적 아이디어를 보다 형식적인 방법으로 기호화하는 것을 도왔다. 프로그래밍 과정에서 학생들의 인지적 모니터링 전략을 발달시키기 위해서는 교사의 안내적 교수법이 필수적이다. 수학교수-학습에 프로그래밍 활동을 효과적으로 적용하기 위해서는 다양한 사례연구를 통한 교수-학습 방법에 관한 연구가 이루어져야 할 것이며, 이에 필요한 수업 교재 및 다양한 탐구형 활동지가 개발되어야 할 것이다. 두리틀은 프로그래밍 환경이 수학교육에 직접적으로 활용되기에는 아직 많은 제한점을 수반하고 있지만, 기존의 프로그래밍보다 더 다양하고 풍부한 마이크로월드(Microworld)의 환경을 제공해 줄 수 있다는 점에서 큰 의미를 가진다.

참 고 문 헌

김경미 (2004). 객체지향형 교육용 프로그래밍 언어 '두리틀(Dolittle)'의 수학교육 활용. 대한수학교육학회 수학교육학논총 **25**, pp.469-488.

김남희 (1997). 변수 개념의 교수학적 분석 및 학습 지도 방향 탐색, 서울대학교 대학원 박사학위 논문.

백영균 · 우인상 (1994). LOGO 프로그래밍의 수업방법이 문제해결력에 미치는 효과에 관한 연구. 교육공학연구 **9(1)**, pp.73-90.

조미옥 (1991). LOGO 프로그래밍의 안내적 교수법을 통한 인지적 모니터링 전략의 발달. 교육공학연구 **7(1)**, pp.161-180.

조한혁 (1991). 교육용 언어의 설계에 대한 연구. 서울대학교 사대논총 **43**.

황우형 · 김경미 (2004). 객체지향형 교육용 프로그래밍 언어 '두리틀(Dolittle)'의 수학 교수-학습 활용 방안. 한국수학교육학회지 시리즈 E <수학교육 논문집> **19(1)**. pp.215-240.

兼宗進(가네무네) (2003). *教育利用を目的としたオブジェクト指向言語の研究*(교육이용을 목적으로 한 오브젝트 지향 언어의 연구). 쓰쿠바대학 박사학위논문.

ACM (2003). *A Model Curriculum for K-12 Computer Science.* Final Report of the ACM K-12 Education Task Force Curriculum Committee. http://acm.org/education/k12.

Brown, A. L. (1983). Learning to learn how to read. In J. Langer and T. Smith Burke(Eds.), *Reader meets author, bridging the gap : A psycholinguistic and social linguistic perspective.* Newark, NJ : Dell.

Clements, D. H. (1987). Longitudinal study of the effects of Logo programming on cognitive abilities and achievement. Journal of Educational Computing Research, **3(1)**, pp.73-94.

Clements, D. H. (1990). Metacomponential Development in a Logo Programming Envitonment. Journal of Educational Psychology, **82(1)**, pp.141-149.

Feuerstein, R. (1979). *The dynamic assessment of retarded performers : The learning potential assessment device, theory, instruments, and techniques.* Baltimore, MD : University Park Press.

George Lukas, Joan Lukas (1986). LOGO*: Principles, Programming, & Projects.* California : Brooks/Cole.

Harvey, V. (1982). Why Logo? *Byte*, **7(8)**, pp.163-171.

Judith Gal-Ezer, David Harel (1999). What (Else) Should CS Educators Know?. *Communications of the ACM* **41(9)**, pp.77-84.

Kris Howell (2003). First Computer Languages. *JCSC* **18(4)**, pp.317-331.

Mibrandt, G. (1995). Using Problem solving to Teach a Programming Language. *Learning and Leading with Technology*, pp.27-31.

Missiuna, C., Hunter, J., Kemp, T., & Hyslop, I. (1987). Development and evaluation of the "thinking with Logo curriculum." *Technical Report No.* 143. Edmonton, Alberta. ERIC ED 287453.

Papert, Symour (1980). Mindstorm : Children, Computers, and Powerful ideas. New York, Basic Books. 류희찬 역, *로고와 아동*.

Salomon, G., & Perkins, D. N. (1987). Transfer of cognitive skills from programming : When and how? *Journal of Educational Computing Research*, **3(2)**, pp.149-169.

Samuels, M. (1986). *Instrumental Enrichment with learning disabled students.* Paper presented at the International Association for Children with Learning Disablities Meeting, Edmonton, Alberta.

Seidman, R. J. (1987). *Research on teaching and learning computer programming symposium.* Paper presented to AERA, Washington, DC. ERIC ED 287442.

Singley, M. K., & Anderson, J. R. (1989). *The transfer of cognitive skill.* Cambridge, MA : Harvard University Press.

Sternberg, R. J. (1984). A contextualist view of the nature of intelligence. *International Journal of Psychology*, **19**, pp.307-334.

Sternberg, R. J. (1987). Teaching intelligence : The application of cognitive psychology to the improvement of intellectual skills. In J.B. Baron and R.J. Sternberg(Eds.), *Teaching thinking skills : Theory and practice.* New York, NY : W.H. Freeman and Company.

Swan, K., & Black, J. B. (1989). Logo programming and the teaching and learning of problem solving. *CCT Report* 89-1, Columbia University.

Ursula Wolz & Edward Conjura (1994). Integrating Mathematics and Programming into A three tired Model for Computing Science Education. *SIGSCE(Special Interest Group on computer Science) Bulletin ACM,* pp.223-227.

Vicki L. Almstrum, Orit Hazzan, David Ginat & Tom Morley (2002). Import and Export to/from Computing Science Education : The Case of Mathematics Education Research. *ACM SIGCSE* **34(3)**, pp.193-194.