韓國數學敎育學會誌 시리즈 A <數學敎育>
1999. 5. 제 38권, 제 1호, 15-35.

J. Korea Soc. Math. Ed. Ser. A: The Mathematical Education
May. 1999, Vol. 38, No. 1, 15-35.

로고(LOGO) 언어의 중등수학교육 활용방안*

황 우 형 (고려대학교)

I. 서 론

1. 연구의 목적

수학을 가르치고 배우는데 지필만을 사용하는 시대는 서서히 그 막을 내리고 있다. 그러나 우리 나라 수학교육은 아직도 지필에 주로 의존하고 있으며, 학생들은 소리 없는 음악수업을 받는 것과 비유될 수 있는 구체적인 모델 없는 수학개념을 학습하고 있다. 수학교육 선진국에서는 이미 오래 전부터 각종 교육보조자료를 사용해서 수학적 개념을 모델로 만들어서 소개하고 있으며 공학기술의 발달로 컴퓨터와 계산기를 사용하는 수업을 연구·개발해서 교육현장에서 널리 사용하고 있다. 미국 NCTM의 "Standards"에서도 기술 공학이 수학과 수학의 용도를 변화시킨다고 생각하고 있으며 다음과 같이 주장하고 있다. 1) 모든 학생은 언제든지 적절한 계산기를 사용할 수 있어야 하며, 2) 시범을 목적으로 모든 교실에서 컴퓨터가 사용가능 해야 하고, 3) 모든 학생은 개인 또는 그룹별 학습을 위해서 컴퓨터 사용이 가능해야 하며, 4) 학생들은 문제를 탐구하고 해결하기 위해, 정보를 처리하고 계산을 수행하는 도구로서 컴퓨터를 사용할 수 있어야 한다.

우리 나라에서도 일부교사들이 교육용 컴퓨터 프로그램을 만들고 있으나, 아직은 초보단계로 이 가운데는 교과서를 컴퓨터 스크린에 옮겨놓은 것에 지나지 않은 프로그램도 많이 있다. 더욱이 중등학교 수준에서는 이에 대한 연구가 초등학교 수준에 비해서 상대적으로 빈약한 실정이다.

본 연구의 목적은 중등수학교육에서 컴퓨터의 활용으로, 특히 LOGO 언어를 활용하는 방안을 연구하여 수학교사들이 이를 쉽게 사용하도록 하는 데 있다.

* 이 논문은 1997년도 한국학술진흥재단의 공모과제 연구비에 의해서 연구되었음.

2. 이론적 배경

1) LOGO의 기원과 배경

LOGO가 어떻게 만들어졌고, 어떻게 작동되며, 실제 교육에 어떻게 영향을 미칠 수 있는지 그리고 다른 컴퓨터 언어와 비교해서 장점은 무엇인지 등은 Papert(1980)의 저서 『Mindstorms: Children, Computers, and Powerful Ideas』에 상세히 설명되어 있다. 이 책을 번역한 백영균과 류희찬(1990)은 LOGO의 교육적 핵심요소는 자기의 사고를 의식화시킨다는 점과 아동들에게 인위적이면서도 자연스러운 학습환경을 제공한다는 점이라고 했다. 여기서 자기의 사고를 의식화한다는 것은 Piget가 규명한 발달의 메카니즘인 반영적 추상화와 유사하며, 이는 물체에 대한 행동의 결과로부터의 추상화를 말한다.

백영균과 류희찬(1990) 또한 LOGO가 교육에 미치는 영향을 다음의 5가지로 정리했다. 첫째, 거북 기하로서의 위치이다. Papert는 그의 구성주의 교육관과 자연스러운 컴퓨터 보조 학습환경을 구체화하기 위해서 "거북 기하"를 만든 것이다. 이 기하는 유클리드 기하나 해석 기하와는 다른 행동을 번역해서 얻어지는 새로운 의미의 기하이다.

둘째는 문제해결을 향상시키는 도구로서의 위치로, LOGO 언어로 프로그램을 만드는 자체가 문제해결 과정이라는 것이다. 목표를 인식하고, 프로그램을 설계하고, 이를 작성해서 실행해보고, 그 결과의 오류를 수정하는 것이 Polya의 문제해결단계(문제의 이해, 계획, 실행, 반성)와 유사하다.

셋째는 직관력을 향상시킬 수 있다는 것으로, 현재 학교 현장에서 이루어지지 않고 있는 직관력을 LOGO를 통해서 발달시킬 수 있다는 것이다. 즉, 자신이 만든 프로그램이 계획했던 것과 다른 경우 이를 직접 "눈"으로 확인한 후 이를 수정할 수 있다.

넷째로 흥미유발과 학습에 대한 태도를 개선하는데

도움이 될 수 있다. LOGO의 교육적 장점은 학생들이 컴퓨터내의 가상적 대상인 거북이에게 어떤 것을 지도한다는 개념으로, 이는 컴퓨터에게 지식의 확장과정을 가르치는 셈이며 따라서 지식이란 "우리가 만들어간다"라는 인식을 분명히 할 수 있고, 학생들이 능동적으로 수학적 지식을 구성해 감으로써 수학학습에 흥미를 갖을 수 있다.

마지막으로 사고유발을 시킨다는 점을 들고 있다. 특히 오류를 발견하는 것이 중요한데, 그 이유는 이를 통해서 학습하기 때문이다. 오류는 예상하지 못했던 곳에서 발생하기 때문에 학생의 흥미를 끌 수 있고, 이를 제거하기 위해서는 어떤 조치를 반드시 취해야 한다.

류희찬(1994)은 수학교육이 다음과 같은 다섯 가지로 뒷받침된다고 주장했다. 이는 구성주의, 사회적 상호주의, 수학화, 정보화 시대의 수학적 소양의 변화, 모든 학생을 대상으로 하는 수학교육이다. LOGO는 이러한 다섯 가지 원리에 충실한 수학교육을 가능하게 해주는 교육매체인 것이다. 이 논문에서 류희찬(1994)은 LOGO의 특성을 반영하는 LOGO 학습환경을 구조화하기 위해서 다음과 같은 6단계의 수업 모델을 고려해 보았다. 이는 준비단계, 구체적 행동 및 행동의 토론, 행동의 표현, 프로그래밍 및 진행, 오류수정, 확장 및 일반화이다.

2) 컴퓨터의 활용방법과 LOGO

Taylor(1980)는 수업에서 컴퓨터의 활용 방법을 교사(tutor), 수업대상(tutee), 도구(tool)의 세 가지로 분류해서 제시하였다. LOGO는 이중에서 수업대상(tutee)에 해당한다. 이 방법은 학생과 컴퓨터의 역할이 바뀐 경우로, 학생이 컴퓨터에게 원하는 활동을 하도록 가르친다.

Jensen, R.J. et. al(1993)은 컴퓨터를 수업대상(Tutee)으로 사용할 것을 강조하며 다음과 같이 주장하고 있다. 첫째, 프로그래밍 하는 것을 배우는 것은 그 차체가 매우 중요한 것으로 교육과정에 포함되어야 한다. 둘째, 학생들은 프로그램 하는 방법을 배우는 과정에서 통찰력을 얻으며, 다른 학습상황에 응용할 수 있는 지식을 얻게 된다. 셋째, 구체적인 수학적 절차를 프로그램으로 만드는 과정에서 학생들은 그 절차를 더욱 세밀하게 분석하게 되고, 그 결과 수학적으로 더 깊이 이해하게 된다. Jensen, R.J. et. al(1993)은 또한 수학에서 절차적인 기능의 교육, 수학적인 개념의 교육, 문제해결과 수학적 추론을 가르치는데 컴퓨터의 역할을 강조하며 이중에서 수학적 개념을 교육하고 문제해결 능력을 신장하는 예로 LOGO를 소개하고 있다.

3) 컴퓨터의 사용과 교사의 역할

LOGO를 비롯한 다양한 컴퓨터 소프트웨어를 수학시간에 활용하는 것은 교사에게 많은 변화를 요구하게 된다. 특히 LOGO를 중등학생에게 가르칠 때 중점을 두어야할 것은 프로그램 자체를 만드는 것과 자신이 만든 프로그램으로 새로운 수학의 세계를 탐구하는 것이다. 프로그램을 만드는 단계에서는 가능하면 최소의 정보와 아이디어를 제공하고 이를 활용해서 프로그램을 만들게 하며, 일단 만든 프로그램으로 탐구할 수 있는 상황을 만들어주는 것이 교사의 역할이다. 전통적인 수학시간과 다른 것은 학생이 생각하는 시간을 충분히 주는 것으로 어떤 과제에 어느 정도의 생각할 수 있는 시간을 할애해야 하고, 어떤 것을 숙제로 내어주며, 이때 어느 정도의 시간을 주는 것이 적절한지 결정해야 하는 것도 교사의 몫이다. Williams(1989)는 교사가 컴퓨터를 사용하는 수업시간에 학급을 효과적으로 운용하는 방법을 다음과 같이 제시하고 있다.

1. 소프트웨어 사용방법을 구두와 글로 알려준다.
2. 학생들이 컴퓨터를 켜고 끄는 방법을 비롯하여 이를 안전하게 다루는 방법을 확실히 알게 한다.
3. 컴퓨터와 함께 사용할 수 있는 다른 방법도 강구한다. 차트나 카드 등이 이에 해당된다.
4. 컴퓨터 실습을 하는 동안 학생들이 교사에게 질문할 수 있는 방법을 강구한다.
5. 컴퓨터를 일종의 "상"으로 사용해서는 안된다. 이는 컴퓨터를 사용하기 싫어하는 학생에게 불공평한 처사이고, 이를 통해서 컴퓨터는 우수한 학생만을 위한 것이라는 잘못된 생각을 갖게 할 수 있다.
6. 소프트웨어를 시연해 보인다.
7. 소프트웨어를 사용하는 시간을 확인한다.
8. 이러한 수업에 도움을 줄 수 있는 학생과 교사가 있는지 알아본다.

3. 연구의 중요성

LOGO 언어는 초기화면의 중앙에 나타나는 거북이(TURTLE)를 사용해서 몇 가지 명령어만 습득하고 나면 쉽게 도형을 그릴 수 있기 때문에 초등학교 수준의 수학을 가르치는데 주로 연구되고 사용되었다. 그러나 LOGO 언어는 그 고유의 특성상 중등수준의 수학에도 이에 대한 활용도가 크며, 이에 대한 연구는 컴퓨터를 수학학습에 활용하는 새로운 방향을 제시하는 계기가 되리라 생각한다.

특히 LOGO 언어는 명령어자체가 배우기 쉽기 때문에 다른 컴퓨터 언어와 비교해서 언어자체를 배우는 시간을 절약 할 수 있으며, 간단히 배운 명령어로 여러 가지 수학적 개념을 탐구할 수 있을 뿐만 아니라, 학생 스스로 의도하는 프로그램을 직접 만들 수도 있다. LOGO를 사용하는 학습방안은 음악시간에 악기를 통해서 음악을 배우듯이 수학시간에 사용할 수 있는 좋은 수학적인 악기가 되어서 학생 스스로 수학적 개념을 이해하고 탐구하며, 수학에 흥미를 가질 수 있는 좋은 기회를 제공할 수 있다. 특히 제6차 교육과정과 7차 교육과정에서 컴퓨터와 계산기의 활용을 적극 권장하고 있으나 이에 대한 구체적인 지침이 없는 시점에서 LOGO의 중등수학의 활용에 관한 연구는 수학교사들에게 하나의 방향을 제시해 줄 것으로 생각된다.

II. 관련된 연구

LOGO는 지난 25년간 연구되어 왔으며, 컴퓨터를 수학교육에 활용하려는 노력이 시작된 시기부터 여러 가지 교육용 소프트웨어 중에서 가장 많은 관심을 가졌던 컴퓨터 언어 중 하나이다. LOGO와 관련된 여러 연구 중에서 최근에 이루어진 연구를 요약해 보면 다음과 같다.

Schuyten & Valcke(1990)은 LOGO 언어와 교사교육(Teacher Education in LOGO-Based Environments)라는 제목으로 책을 편집했는데, 이 책의 서론 부분에서는 LOGO 언어의 특징, LOGO의 수업에 활용, 수학교육에서 LOGO의 영향, 구성주의, LOGO의 잠재력 등을 다루고 있다. 각 장에는 벨기에, 그리이스, 아일랜드, 네덜란드, 포르투갈, 스페인 등의 유럽국가에서 공동 연구한 연구결과가 각각 언급되어 있으며, 각 나라에서는 자신들이 시행하고 있는 교사 교육 중 LOGO와 관련된 부분에 대해서 설명하고 있다. Kern & Mauk(1990)는 LOGO를 사용해서 Fractal을 만드는 프로그램을 소개했고, Lemerise(1990)는 LOGO를 정규수학교육과정에서 사용할 수 있는지 시도해 보기도 했다.

Swan & Black(1990)는 초등학교부터 대학원까지의 학생을 연구대상으로 LOGO 언어가 문제해결 능력을 개발하는데 미치는 영향에 대해서 연구했는데, 연구결과 단계적으로 목표 세우기, 문제를 모델링 하기, 체계적인 시행착오 실시 등의 문제해결전략에 긍정적인 영향을 미치는 것을 알아냈다. Yusuf(1990)는 33명의 9학년(중 3) 학생들을 두 집단으로 나누어서 연구를 실시했다. 한 집단은 LOGO 프로그램, LOGO 개인교수 프로그램, 학급활동지, 숙제 등을 사용해서 가르쳤고, 다른 그룹은 전통적인 교수법으로 같은 내용을 공부했다. 연구결과 LOGO를 사용했던 집단이 기하에 대한 태도뿐만 아니라 성취도도 높아졌다는 것을 알아내었다.

Grandgenett(1991)은 학생들의 추론능력을 계발하기 위해서 LOGO를 사용했는데, Sternberg의 네 가지단계(encode, infer, map, apply)를 사용했다. Olive(1991)는 LOGO 프로그램을 만드는 것과 기하학을 이해하는 것의 상관관계를 Van Hiele 수준, Skemp의 수학적인 이해 모델 등을 기준으로 연구했다. 연구 결과 LOGO프로그램을 잘 만드는 것이 주어진 과제의 기하학적인 부분을 이해하는데 필요조건이지만 충분조건은 아니라는 것을 알아냈다. McAllister(1991)는 영재학생들을 대상으로 이들이 LOGO 언어를 배울 때 사용하는 문제해결 전략에 대해서 연구했다. 연구결과에 따르면 학생들이 사용한 전략은 과제가 요구하는 내용, 프로그램을 만드는 능력에 달려있었다. Maddux(1992)는 LOGO 언어를 포함해서 학교에서 컴퓨터를 사용하는 것에 대한 장점과 단점에 대해서 논의했으며, Nevile(1992)이 편집한 "LOGO and Mathematics Education LME5"에서는 LOGO의 소프트웨어나 컴퓨터 하드웨어의 변화보다는 LOGO의 수학적이고 교육학적인 문제에 초점을 맞추어서 상세히 논의한 바 있다. Denenberg(1993)는 문제해결을 가르치는데 LOGO를 사용할 것을 제안하고 있으며, LOGO를 사용하는 여러 가지 프로그램의 예도 제시한 바 있다.

국내에서도 LOGO에 관한 연구가 상당수 있었다. 류희찬과 이지요(1993)는 LOGO 언어를 통한 시각화의 중요성을 강조한 바 있다. 그 동안 시각화가 소홀히 다루어진 이유를 유클리드 이후의 수학이 보편적인 진리 추구의 성격을 띠면서 시각화는 설명을 위한 보조적인 위치로 전락한 것으로 보았고, 또 다른 이유는 적절한 도구가 갖추어지지 않았기 때문으로 생각했다. 류희찬과 이지요(1993)는 시각화를 위한 LOGO 환경을 6가지로 분류해서 설명했는데, 이는 시각화를 위한 LOGO 환경, 탐구할 수 있는 환경, 저학년부터 배우기 쉬운 환경, 자신의 행동을 의식화시키는 환경, 재미있는 환경, 형식적 수학수업을 위한 잠재력을 향상시킬 수 있는 환경이다. 이들은 또한 LOGO 활동의 실례로 각과 다각형, 테슬레이션, 프랙탈, 코엡 그래프 등을 들었으며, LOGO는 학생들이 쉽게 다룰 수 있는 그래픽 프로그래밍 언어라는 점에서 LOGO 마이크로 세계는 시각화를 위한 좋은 환경이라고 결론을 맺었다.

김수환과 이재학(1992)은 중학교 2학년 학생들을 대상으로 LOGO 프로그래밍 활동을 통한 논리-수학적 경험이 인지발달을 촉진하는지에 대해서 연구하였다. 실험집단과 통제집단을 각각 35명씩 선정하여 실험집단은 3주에 걸쳐서 주당 3-5일씩 하루에 45분씩 12일 동안 LOGO 프로그래밍 활동을 했다. Tobin과 Capie가 개발한 TOLT(Test of Logical thinking)을 번안한 도구로 사전 검사와 사후 검사를 실시한 결과 LOGO 프로그래밍 활동을 통해서 실험집단 학생들의 논리적 사고력이 향상된 것으로 나타났다.

III. 연구내용

1) LOGO의 기본명령어

이 소단원에서는 LOGO 언어의 가장 기본이 되는 명령어를 소개하려고 한다. 더욱 자세한 내용을 알기 위해서는 시중에 나와 있는 LOGO에 관한 참고문헌을 참조하면 도움이 될 것으로 생각된다. LOGO 프로그램의 초기화면이 뜨면 거북이가 보이는 그래픽 화면부분과 그 아래쪽에 명령부가 나타나는데, 이 명령부에서 명령을 하면 그 명령에 따라서 거북이가 움직인다.

거북이를 앞쪽으로 움직이는 명령어는 FORWARD (FD)로 이 명령어를 사용해서 거북이를 앞으로 움직이게 할 수 있다. "FORWARD 50"을 명령부에 쓰고 "ENTER"를 누르면 거북이는 앞으로 50만큼 이동한다. BACKWARD(BK)를 사용하면 거북이를 뒤로 움직일 수 있다. "FORWARD 30"을 명령하면 뒤로 30만큼 이동한다. 이때 FD와 BK의 명령어를 활용해서 정수의 연산의 예를 보여줄 수 있다. 예를 들어, FD 50은 거북이가 전진하여 길이가 50인 선을 그리므로 이는 +50을 나타낸다고 볼 수 있으며, BK 50은 거북이가 후진하면서 길이가 50인 선을 그리므로 이는 -50을 나타낸다고 생각할 수 있다. 이때 BK -50 등의 명령을 실행하도록 하면 음수에 음수를 곱하면 양수가 되는 하나의 예를 스스로 알 수 있는 계기를 마련해 줄 수 있다.

거북이의 방향을 바꿀 때 사용하는 명령어는 RIGHT (RT)로 거북이의 방향을 오른쪽으로 바꾼다. "RIGHT 90"을 명령하면 90도만큼 오른쪽으로 이동한다. 이와 비슷하게 LEFT(LT)를 사용하면 거북이의 방향을 왼쪽으로 바꿀 수 있다. "LEFT 90"을 명령하면 90도만큼 왼쪽으로 방향을 바꾼다. RT와 LT의 명령어에 다양한 각도를 주고 위의 FD와 LT 명령어와 함께 사용하면 여러 가지 모양이나 도형을 만들 수 있다. FD와 BK, 그리고 RT와 LT 명령어와 익숙해지도록 하기 위해서 학생들 자신이 고안한 모양이나 도형을 4가지 명령어(FD, BK, RT, LT)로 만들어 보는 것은 바람직한 활동이라 할 수 있다. 그런데 학생들이 그리고자하는 그림이 복잡해지고 여러 개의 모양을 한 화면에 그리고자하면 위의 네 가지 명령어로는 한계를 느끼게 된다. 이때 PU(PEN UP), PD (PEN DOWN) 명령어를 도입한다.

위에서 설명한 FORWARD나 BACKWARD를 사용해서 명령하면 거북이는 움직이면서 그 자취를 남긴다. 그런데 거북이는 움직이고 싶지만 자취를 남기고 싶지 않을 때는 PU(PEN UP) 명령어를 사용한다. PU을 명령부에 쓴 후 ENTER 키를 누르면 이후에는 거북이를 움직여도 거북이만 움직이고 자취가 남지 않는다. 그리고 나서 다시 자취를 남기기를 원하면 PD(PEN DOWN) 명령을 사용한다. 즉, PD를 명령부에 쓰고 ENTER키를 누르면 그 후부터는 거북이의 자취가 다시 남는다. 그러므로 어떤 그림이나 도형을 그리고 나서 그 옆에 다른 그림을

그리고 싶을 때는 PU을 명령부에 쓰고 ENTER를 누른 후 FD, BK, RT, LT 등의 명령어를 사용해서 거북이를 이동시킨다. 그러면 거북이는 이동하지만 자취는 남지 않는다. 그리고 나서 다른 그림을 그리고 싶으면 PD을 명령부에 쓰고 ENTER를 누르면 그 다음부터 명령어로 거북이를 움직이면 거북이가 움직이면서 자취를 남긴다.

명령어 REPEAT는 같은 명령을 여러 번 반복할 경우 사용한다. 한 변이 30인 정사각형을 명령어로 그리려면 "FD 30 RT 90 FD 30 RT 90 FD 30 RT 90 FD 30 RT 90"이라고 명령해야 한다. 이때 명령어의 순서를 살펴보면 FD 30 RT 90 이 4번 반복되는 것을 알 수 있다. LOGO 명령어에는 이와 같이 반복되는 경우 과정을 간단히 쓸 수 있도록 "REPEAT"라는 명령어가 있다. 위의 과정을 REPEAT를 사용해서 나타내면 "REPEAT 4 [FD 30 RT 90]"와 같이 간단히 나타낼 수 있다.

LOGO 언어에는 여러 가지 과정이나 그려진 그림을 저장하는 기능이 있다. 어떤 과정이나 그림을 그린 후 이를 저장하고 싶으면 NAMEPAGE (NP ")라는 명령어를 사용하면 된다. LOGO 초기화면의 뒷면(FLIP SIDE)에 어떤 과정을 작성한 후 NP "PROGRAM1을 명령부에 쓰고 "ENTER"키를 누르면 PROGRAM1이라는 이름으로 작성된 과정이 저장된다. 초기 화면에서 FLIP SIDE로 가려면 CTR+F 키를 누르면 된다. 이때 TO로 시작해서 END로 끝나는 프로그램을 여러 개 만들 수 있는데 이에 대한 예는 아래에 나와있다.

CT(CLEAR TEXT)는 CT를 명령부에 쓰고 "ENTER" 키를 누르면 화면부에 나타났던 모든 결과가 지워지며, HT(HIDE TURTLE)를 명령부에 쓰고 "ENTER" 키를 누르면 화면부에 나타났던 거북이가 사라진다. 이때 명령어를 사용해서 거북이를 움직일 때 거북이의 모습은 보이지 않지만 자취는 남으므로 그림을 그릴 수 있다.

이제는 간단한 명령이 아닌 하나의 프로그램을 만드는 방법에 대해서 알아보자. 모든 LOGO프로그램은 TO로 시작해서 END로 끝난다. 예를 들어, 한 변의 길이가 50인 정사각형을 그리는 절차를 만들기 위해서는 다음과 같이 TO와 END를 사용해서 만들면 된다.

```
TO SQUARE
REPEAT 4[FD 50 RT 90]
END
```

위의 절차를 뒷면(FLIP SIDE)에 만들고 다시 앞면으로 돌아와서 명령부에 SQUARE를 쓰고 나서 "ENTER"를 누르면 위의 절차가 실행되어서 거북이는 정사각형을 그리게 된다.

2) LOGO의 기본 프로그램

LOGO 언어를 통해서 학생들의 수학적인 사고를 개발시키는 두 가지 중요한 LOGO의 기능이 있는데 이것은 변수(variable)기능과 재귀과정(recursive procedure) 기능이라고 할 수 있다. LOGO의 변수기능을 이용해서 다음의 그림을 그리는 절차를 만드는 것은 변수의 개념을 이해하는데 많은 도움을 줄 수 있다. 위에서 한 변의 길이가 50인 정사각형을 그리는 절차에 대해서는 이미 알아보았다. 그러면 이 프로그램은 정사각형의 크기를 조절하기 위해서는 프로그램을 수정하는 방법 밖에 없다. 정사각형의 크기를 마음대로 조절하기 위해서는 어떻게 프로그램을 만들어야하는지 알아보겠다.

(1) 변수(variable) 기능의 활용

```
TO SQUARE
REPEAT 4[FD 50 RT 90]
END
```

위의 SQUARE 프로그램에서 정사각형의 크기를 정하는 부분은 FD 다음에 나오는 숫자 50이다. 즉, 한 변이 50인 정사각형을 그리게 된다. 그러므로 정사각형의 크기를 조절하려면 FD 다음에 나오는 숫자를 변수(variable)로 만들어야 한다. 어떤 정해진 숫자를 변수로 만드는 방법은 TO SQUARE 다음에 한 칸 띄운 후 ":"를 입력하고 아무 이름이든지 프로그램을 만드는 사람이 원하는 변수의 이름을 쓰면 된다. 아래의 경우에는 변수의 이름을 SIDE로 사용하고 있다. 그 다음에 FD 50을 같은 변수 SIDE를 사용해서 FD :SIDE 로 바꾸면 된다.

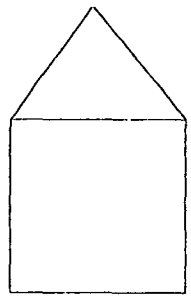
```
TO SQUARE :SIDE
REPEAT 4[FD :SIDE RT 90]
END
```

위의 프로그램을 만들고 나서, CTL+F를 동시에 눌러서 원래 화면으로 돌아온 후 명령부에 SQUARE 50을 쓰고 "ENTER"를 누르면 길이가 50인 정사각형이 그려지고, 명령부에 SQUARE 30을 쓰고 "ENTER"를 누르면 길이가 30인 정사각형이 그려진다. 이때 RT 다음에 나오는 90을 변수로 만들면 어떤 프로그램이 될 것인지 학생에게 예상해보도록 하는 것은 좋은 활동이다. 이때 각을 변수로 만들려면 변수가 하나 더 필요하게 된다. 새로운 변수를 ANGLE이라고 부르기로 하면 크기와 각을 바꾸는 프로그램은 다음과 같다. 즉, 변수는 하나 이상을 만들 수 있다.

```
TO SQUARE :SIDE :ANGLE
REPEAT 4[FD :SIDE RT :ANGLE]
END
```

이 프로그램을 실행시키기 위해서는 명령부에 SQUARE 50 90과 같이 SQUARE 뒤에 숫자를 두 개 입력하면 된다. 이때 두 수 사이는 한 칸을 띄어야 한다.

다음은 변수를 이용해서 "HOUSE"라는 이름으로 아래와 같은 집을 한 채 그려보려고 한다. 다시 말하면, 원하는 크기로 집의 크기를 조절할 수 있도록 프로그램을 만드는 것이다. 이때 "HOUSE"를 그리기 위해서는 "SQUARE"와 "TRIANGLE"이라는 하위 프로그램을 먼저 만들어야 하는데, 이 하위 프로그램들도 변수를 포함하고 있어야 한다. 그 이유에 대해서 학생들에게 묻는 것도 학생들이 변수의 사용을 이해하는데 질문이 될 수 있다.



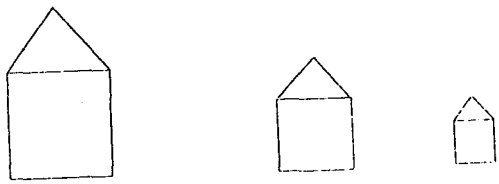
<그림 1>

```
TO SQUARE :SIDE
REPEAT 4[FD :SIDE RT 90]
END

TO TRIANGLE :SIDE
REPEAT 3[FD :SIDE RT 120]
END

TO HOUSE :SIDE
SQUARE :SIDE
FD :SIDE RT 30
TRIANGLE :SIDE
LT 30 BK :SIDE
HT
END
```

프로그램 "HOUSE"의 마지막 부분에 LT 30 BK :SIDE을 첨가한 이유를 나중에 "VILLAGE"를 만드는 프로그램 연관지어서 학생들에게 묻는 것도 프로그램을 만드는 방법을 이해하는데 도움이 된다. 변수를 갖는 프로그램 "HOUSE"를 그리고 나면 이를 이용해서 아래와 같은 마을(VILLAGE)을 그릴 수 있다. 이때 유념해야 할 것은 VILLAGE의 변수를 변화시킴에 따라서 집의 크기도 변하고 마을전체의 크기도 변하도록 프로그램을 만들어야 한다는 것이다. 이 프로그램을 만들 때 고려해야 하는 것은 집 사이의 간격이 SQUARE의 한 변의 길이와 같다는 것과 SQUARE의 크기가 10씩 줄어든다는 것이다. VILLAGE를 만들려면 HOUSE를 하위 프로그램으로 사용해서 집이 점점 작아지게 해야하고, 집을 한 채 그린 후 다음 집을 그리기 위해서 이동해야 하므로, 이 이동 과정을 MOVE라는 하위 프로그램으로 만들어야한다. 위의 내용을 설명한 후 학생들이 스스로 프로그램을 만들 수 있도록 충분한 시간을 준다.



<그림 2>

```
TO MOVE :SIDE
PU RT 90 FD :SIDE * 3 LT 90 PD
END

TO VILLAGE :SIDE
HOUSE :SIDE MOVE : SIDE
HOUSE :SIDE - 10 MOVE :SIDE - 10
HOUSE :SIDE - 20
END
```

위에서 설명한 바와 같이 한 프로그램에서 변수를 두 개 이상 동시에 사용할 수 있다. 사각형의 가로와 세로의 크기를 마음대로 조절할 수 있는 절차를 만들기 위해서는 변수를 두 개 주면 된다. 이때 이 과정도 변수가 있는 정사각형 만드는 프로그램을 참고로 하도록 해서 학생들이 스스로 만들어 보도록 하고, 이를 만든 후 원하는 사각형을 만들 수 있는지 확인해 보도록 한다.

```
TO RECTANGLE :WIDTH :LENGTH
FD :WIDTH  RT 90
FD :LENGTH  RT 90
FD :WIDTH  RT 90
FD :LENGTH  RT 90
END
```

또는

```
TO RECTANGLE :WIDTH :LENGTH
REPEAT 2 [FD :WIDTH RT 90 FD :LENGTH RT 90]
END
```

지금까지는 원하는 도형이나 그림을 그리기 위해서 LOGO 언어를 이용해서 프로그램을 만들었다. 그런데 이와 반대로 변수를 포함하는 프로그램을 보고 그 프로그램이 그리는 도형을 추론해 보는 것도 학생들의 공간추론 능력 개발하는데 매우 유익한 활동이다. 다음 프로그램을 보고 TRAIN 20 50, DESIGN 20 50, ARCH 50 20 이라고 명령했을 때 그리게 되는 도형을 종이 위에 직접 그려보게 하고, 그 그림이 정확한지 이를 컴퓨터로 확인해보는 것이다. 이 활동에는 변수가 두 개나 등장하고 REPEAT가 나오면 그 수만큼 반복해야 하고 주어진 각도에 따라서 방향을 바꾸어야 하기 때문에 공간추론 능력을 개발하는데 큰 도움이 된다. 이때 하위 프로그램으로 쓰여진 RECTANGLE은 바로 전에 만든 프로그램이다. 여기에 나와있는 세 가지 프로그램은 단지 예를 든 것이고 학생들을 2인 1조가 되도록 짝지어서 각자가 만든 프로그램을 주고받으며 상대가 만든 프로그램을 그림으로 그려보고 이를 비교 확인해 보는 활동으로 이어질 수 있다.

```
TO TRAIN :W :L
RECTANGLE :W :L
RT 90  FD :L  LT 90
RECTANGLE :L :W
END

TO DESIGN :W :L
REPEAT 2 [RECTANGLE :W :L FD :L]
RT 180
REPEAT 2 [RECTANGLE :L :W FD :L]
END

TO ARCH :W :L
RECTANGLE :W :L
FD :W
RECTANGLE :L :W
RT 90  FD :W  RT 90
RECTANGLE :W :L
END
```

LOGO 언어를 사용해서 원의 성질을 접근하는 방법은 다소 특이하다 할 수 있다. 사실 LOGO 언어를 가지고는 원을 그릴 수 없다. 그러나 정삼각형, 정사각형, 정오각형과 같이 점차 증가시키면 정다각형이 원으로 보이기 시작하며, 이러한 정다각형을 가지고 호(弧), 반원, 원의 성질을 탐구해 볼 수 있다. 먼저 학생들에게 정 다각형 만드는 프로그램(RECCIR)을 만들도록 하고 이를 직접 적용해서 원을 만드는 프로그램을 만들도록 하는데, 원은 360도이므로 학생들에게 정 360각형을 만드는 프로그램을 만들어 보게 하고 이를 CIRCLE이라 명명하기로

한다.

정다각형을 만드는 프로그램을 만들기 위해서는 변수를 사용해야 하는데, 변수를 몇 개 사용해야하고 어떻게 사용해야 하는지 학생들에게 생각해 보도록 해야 한다. 어떤 정다각형을 그리려면 외각과 한 변의 길이가 주어져야 하므로 필요한 변수는 2개이다. 일반적으로 정다각형을 만드는 프로그램을 한 번에 만드는 것은 대부분의 학생들에게 쉽지 않으므로 정삼각형(TRIANGLE), 정사각형(RECTANGLE), 정육각형(HEXAGON) 등을 만드는 프로그램을 먼저 만들도록 하고 이 프로그램들을 보고 규칙성을 발견하게 하면 귀납적 추론활동 기회를 제공하게 되고 쉽게 정다각형을 만드는 프로그램을 만들 수 있게 된다.

```
TO TRIANGLE :L
REPEAT 3 [FD :L RT 120]
END

TO RECTANGLE :L
REPEAT 4 [ FD :L RT 90]
END

TO HEXAGON :L
REPEAT 6 [ FD :L RT 60]
END
```

위의 세 가지 프로그램을 비교해보면 원하는 다각형에 따라서 REPEAT 다음의 숫자가 바뀌는 것을 알 수 있다. 그러므로 :A 라는 변수를 하나 더 만들어서 원하는 다각형의 숫자를 입력하면 된다. 이제 남은 문제는 어떻게 다각형의 외각이 입력하는 숫자에 따라서 변하도록 만드는 것이다. 이 문제는 위의 세 가지 프로그램의 RT 다음에 나타나는 각을 자세히 비교해 보면 해결된다. 3각형은 120, 4각형은 90, 6각형은 60이므로, 3×120=4×90=6×60=360이므로, 다각형에 따라서 원하는 각을 만들고자 하면 원하는 다각형의 숫자로 360을 나누면 된다. 이를 확인해 보려면 정오각형(PENTAGON), 정팔각형(OCTAGON) 등의 프로그램을 만들어 보면 된다.

```
TO PENTAGON :L
REPEAT 5 [FD :L RT 360 / 5]
END

TO OCTAGON :L
REPEAT 8 [FD :L RT 360 / 8]
END
```

이제 원하는 다각형을 만드는 프로그램을 만들면 다음과 같이 된다.

```
TO RECCIR :A :L
REPEAT :A [FD :L RT 360 / :A]
END
```

RECCIR는 정다각형을 만드는 프로그램으로, 원을 정360각형으로 생각해서 원(CIRCLE)을 만드는 프로그램을 학생들에게 만들게 한다.

```
TO CIRCLE
REPEAT 360 [FD 1 RT 1]
END
```

위의 절차는 크기가 일정한 원을 그리게 된다. 다음 단계는 학생들에게 크기를 조절할 수 있는 원(VCIRCLE)을 그리는 방법을 생각하게 한다. 위의 CIRCLE 프로그램에서 크기를 조절할 수 있는 부분은 어느 것인가? "REPEAT 360", "FD 1"과 "RT 1"을 변화시키면서 어느 부분이 원의 크기를 조절하는데 사용되는지를 학생들이 직접 알아보도록 한다. 즉 FD 1, FD 0.5, FD 2 등을 위의 프로그램에 대입해서 원의 크기에 영향을 주는지 알아보는 것이다. "FD" 다음의 숫자가 원의 크기와 관련이 있으므로 학생들에게 이를 변수로 만들어서 크기를 조절할 수 있는 원(VCIRCLE)을 만들게 한다.

```
TO VCIRCLE :S
REPEAT 360 [ FD :S RT 1]
END
```

위의 프로그램을 만들고 나서 이것으로 크기가 다른 원을 그릴 수 있는지 VCIRCLE 0.5, VCIRCLE 1, VCIR-

CLE 1.5 등을 실행해서 알아볼 수 있다.

그런데, 위의 VCIRCLE에서 원의 크기는 조절할 수 있지만 VCIRCLE 1에서 1이 의미하는 것은 VCIRCLE 0.5보다 크고 VCIRCLE 1.5보다 작다는 것뿐이다. 원하는 크기의 원을 정확히 얻으려면 다른 프로그램을 만들어야 한다. 위의 프로그램에서 여러 가지로 원의 크기를 조절할 수 있지만 원하는 반지름을 갖는 원(RCIRCLE)을 그리는 방법을 학생들에게 만들어보게 할 수 있다. 다시 말하면, RCIRCLE 30 이라고 실행시키면 반지름이 30인 원을 그려야 한다. 여기에서 30이 의미하는 것은 LOGO에서 FD 30을 실행했을 때 그리는 직선의 길이를 말한다.

RCIRCLE을 만들 때 VCIRCLE을 활용하면 간단하게 만들 수 있다. VCIRCLE은 원의 크기를 조절할 수 있는 절차로 VCIRCLE 1은 FD 1을 전진한 후에 RT 1 즉, 오른쪽으로 1도 방향을 바꾼다. 이런 식으로 한바퀴 돌아서 원을 만들면 원주는 360(360×1)이 된다. 원주와 반지름의 관계를 식으로 나타내면 $360 \times S=2 \times 3.14159 \times R$ 이 된다. 이 식을 S에 관해서 다시 쓰면, $S=2 \times 3.14159 \times R \div 360$이 된다. 이와 같은 사실을 학생들과 함께 알아보고 이 결과와 VCIRCLE을 하위 프로그램으로 사용하여 RCIRCLE을 만들어보도록 한다.

```
TO RCIRCLE :R
VCIRCLE   2 * 3.14159 * :R / 360
END
```

이 프로그램이 원하는 결과를 보여주는지 알아보기 위해서 RCIRCLE 30, RCIRCLE 50, RCIRCLE 100 등을 실행해서 나타난 원의 반지름을 거북이를 이동시켜서 알아볼 수 있다. 학생들에게 RCIRCLE이 제대로 만들어져 있는지 확인하는 방법에는 어떤 것이 있는지 질문해 볼 수 있다. 즉, RCIRCLE 30을 실행한 후 RT 90 FD 60을 실행했을 때 거북이가 원의 지름만큼 전진하면 된다.

이제는 위의 원을 만드는 절차를 참고로 해서 반원(SEMICIRCLE) 만드는 프로그램을 만들게 하고, 반원 만드는 프로그램을 만든 후에는 반원의 크기를 조절할 수 있는 프로그램(VSEMICIRCLE)도 만들게 한다. 원을 만들었을 때 위의 "CIRCLE"에서 360과 RT 다음의 숫자 1의 관계를 생각해보게 하고 반원 만드는 프로그램을 만들게 한다. 학생들이 반원의 크기를 조절할 수 있는 프로그램(VSEMICIRCLE)을 만들 때는 "VCIRCLE"을 만들 때 사용했던 아이디어를 활용하도록 유도한다.

```
TO SEMICIRCLE
REPEAT 180 [FD 1 RT 1]
END

TO VSEMICIRCLE :SIZE
REPEAT 180 [FD :SIZE RT 1]
END
```

이제는 각의 크기를 조절할 수 있는 호(弧, RARC)를 그리는 절차를 만들고자 한다. 반원을 만드는 절차 "SEMICIRCLE"을 활용하면 원하는 각의 크기를 가진 호를 그릴 수 있다. SEMICIRCLE에서는 반원을 만들기 위해서 360의 반을 취해서 180을 사용했다. 그러므로 각을 조절하려면 어느 부분을 변수로 만들어야 하는지 학생들에게 생각해보도록 한다. 이때는 REPEAT 다음의 수를 변수로 만들면 된다.

```
TO RARC :N
REPEAT :N [FD 1 RT 1]
END
```

프로그램이 원하는대로 만들어졌는지 확인하는 방법에 대해서 학생들에게 질문할 수 있으며, RARC 30, RARC 90, RARC 270 등을 실행하면서 이를 확인해 볼 수 있다. 또한 RARC 절차를 참고해서 각 뿐만 아니라 크기도 원하는대로 그릴 수 있는 프로그램을 만들어 보게 할 수 있다. 원(CIRCLE)에서도 크기는 FD 다음의 숫자 크기에 따라서 변했으므로 변수를 두 개 주면 각과 크기를 원하는대로 그릴 수 있는 절차를 만들 수 있다.

```
TO VRARC :S :N
REPEAT :N [FD :S RT 1]
END
```

"VRARC"는 호를 오른쪽으로 그리게 되는데, 왼쪽으로 그리는 호를 그리는 프로그램(VLARC)을 만들어볼

수도 있다. 방향을 오른쪽으로 돌게 한 것은 RT이므로 이를 LT로 바꾸면 된다.

```
REPEAT :N [FD :S LT 1]
END
```

원과 반원, 그리고 호등을 그리는 프로그램을 만들어 보았는데 이를 활용해서 학생들이 기하에서 사용하는 도형 및 그 외의 원, 반원, 그리고 호등을 포함하는 여러 가지 도형을 그리게 함으로써 공간 추론능력을 향상시킬 수 있다.

(2) 재귀과정 (recursive procudure)의 활용

LOGO 언어의 특징 중에 하나인 재귀(recursive)기능은 변수(variable)를 사용하는 기능과 함께 수학적인 사고력을 개발하는데 도움을 주는 기능이다. LOGO 언어의 재귀 기능을 학생들에게 설명하기 전에 다음과 같은 절차를 주고 어떤 도형을 그리게 되는지 예상하게 하는 것도 좋은 활동이라 할 수 있다.

```
TO CIRC
FD 1 RT 1
CIRC
END
```

위와 같은 프로그램을 실행시키면 거북이가 계속 원을 그리면 돌게 된다. 이때 학생들에게 그렇게 되는 이유를 물어볼 수 있다. 그 이유는, CIRC를 실행시키면 거북이가 앞으로 1만큼 가고 오른쪽으로 1도만큼 회전한 후 다시 CIRC를 만나게 되어서 다시 처음으로 돌아가서 앞으로 1만큼 가고 오른쪽으로 1도만큼 회전하게 되므로 이 과정이 계속된다. 참고적으로, 계속 움직이는 거북이를 멈추게 하려면 CTL+S키를 동시에 누르면 된다.

위와 같은 재귀기능을 사용해서 변의 길이와 각을 조절할 수 있는 POLY, 변의 길이가 일정하게 증가해서 나선형의 모양을 나타내는 POLYSPI 등의 프로그램을 학생들에게 제공하고 변의 길이와 각을 여러 가지로 주면서 나타날 모양을 예상하는 것도 좋은 수학적인 사고활동이라 할 수 있다.

```
TO POLY :SIDE :ANGLE
FD :SIDE RT :ANGLE
POLY :SIDE :ANGLE
END
```

POLY 50 72를 실행시키고 나서 POLY 40 60, POLY 70 120 이 만들게 되는 도형 예상해서 그리게 한 후 이를 실행해서 얻은 도형과 비교하게 할 수 있다. POLY 1 1, POLY 70 144, POLY 80 135, POLY 70 108, POLY 70 160, POLY 50 181 등의 다소 예상하기 어려운 도형들도 위와 같이 예상하도록 하고 나중에 이 프로그램을 실행해서 예상과 다르게 나타나는 경우 어느 부분이 잘못 되었는지 자가진단해 볼 수 있다.

```
TO POLYSPI :SIDE :ANGLE
FD :SIDE RT :ANGLE
POLY :SIDE + 5 :ANGLE
END
```

POLYSPI도 POLY와 마찬가지로 위의 프로그램을 주고 POLYSPI 5 90을 실행해 보도록 한다. 그런데 POLYSPI나 POLY를 실행해 보면 그림을 너무 빠르게 그리기 때문에 어떤 그림을 그렸는지 판단하기 어렵게 되는데 이때에는 SLOWTURTLE 명령어를 명령부에 쓰고 ENTER키를 누르면 그 다음부터 모든 그림을 그릴 때 천천히 그리게 된다. 나중에 다시 빠르게 그리고 싶으면 FASTTURTLE 명령어를 사용하면 된다. POLYSPI 5 90을 실행해 본 후 POLYSPI 5 120, POLYSPI 5 144, POLYSPI 5 117 등이 그리는 모양을 예상해 보도록 할 수 있다.

지금까지의 프로그램에는 재귀과정을 정지하게 할 수 있는 부분이 없었는데, 프로그램을 어느 정도에서 정지시키고 싶으면 IF~[STOP] 명령어를 사용하면 된다. 위의 POLYSPI 프로그램에서 어느 정도에서 거북이가 멈추기를 원한다면 IF :SIDE > 300 [STOP]이라는 명령구를 중간에 첨가하면 된다. 학생들에게 어느 부분에 이 명령구를 넣으면 좋을지 질문해 볼 수 있다.

```
TO POLYSPI :SIDE :ANGLE
IF :SIDE > 300 [STOP]
```

```
FD :SIDE RT :ANGLE
POLY :SIDE + 5 :ANGLE
END
```

참고로, 위의 완성된 프로그램의 :SIDE > 300에서 300의 크기를 적당히 조절해서 프로그램이 정지하는 시점을 조절할 수 있다.

다음에 소개하는 프로그램은 학생들이 탐구할 수 있는 환경을 제공한다. 각의 크기를 입력함에 따라서 만드는 모양을 기준으로 몇 가지 범주로 나눌 수 있는데, 이때 입력하는 숫자가 어떤 영향을 미치는지 탐구해 볼 수 있다.

```
TO INSPI :SIDE :ANGLE
FD :SIDE RT :ANGLE
INSPI :SIDE :ANGLE + 5
END
```

이때 학생들에게 탐구를 시작하게 하기 위해서 다음과 같은 숫자를 입력하도록 하면 도움이 된다. INSPI 10 10, INSPI 4 2, INSPI 5 2.5, INSPI 8 1을 실행시켜보면 나타나는 모양에 따라서 몇 가지의 범주로 나누어 볼 수 있다. 위의 INSPI 프로그램은 각이 5도씩 증가하게 되어 있는데, 변수를 하나 더 첨가해서 증가하는 각도를 원하는 대로 변경해 볼 수도 있고, 이때에는 어떤 범주로 나누어지고 이는 변하는 각도와 어떤 관계가 있는지 탐구해 볼 수 있다.

```
TO INSPI2 :SIDE :ANGLE :INCREMENT
FD :SIDE RT :ANGLE
INSPI :SIDE :ANGLE + :INCREMENT
END
```

3) LOGO의 중등수학활용

지금까지는 기하와 관련된 프로그램을 만들면서 공간추론능력을 개발하는데 초점을 맞추었다면 지금부터는 수론과 수열과 관련된 프로그램들을 만들면서 학생들의 수학적인 사고력을 신장하는데 초점을 맞추고자 한다. 이를 위해서 명령어 PRINT에 대해서 알고 있어야 하는데, PRINT "HELLO, PRINT "HELLO", PRINT "HELLO, PRINT " HELLO, PRINTHELLO, PRINT HELLO, PRINT "HELLO THERE , PRINT "2 + 3, PRINT 2 + 3, PRINT "3 + 2, PRINT 13 - 9 + 5 * 3 등을 실행해 보도록 해서 명령어 PRINT의 사용법을 학생 스스로 알게 한다. 지금부터 소개하는 프로그램은 LOGO에 관한 자료집 등에 수록된 내용으로, 본 연구에서는 이를 학생들에게 어떤 방법으로 소개하는 것이 적절한지 이에 대한 방안을 강구해 보았다.

(1) 약수와 배수

다음과 같은 명령구를 실행시켜서 LOGO에 내장되어 있는 REMAINDER 명령어의 성질을 알게 한다.

```
PRINT REMAINDER 10 3 ---- (화면에 1이 뜬다.)
PRINT REMAINDER 10 2 ---- (화면에 0이 뜬다.)
```

10을 3으로 나누면 나머지가 1이고, 10을 2로 나누면 나머지가 0이므로 화면에 1과 0이 각각 뜬다. 다른 숫자들도 입력해서 이를 확인하도록 한다.

```
PRINT (REMAINDER 10 2) = 0 ------ (화면에 "TRUE"가 뜬다.)
PRINT (REMAINDER 10 3) = 0 ------ (화면에 "FALSE"가 뜬다.)
```

이 경우도 10이 2로 나누어서 떨어지는 경우에는 "TRUE"가 뜨고 10이 3으로 나누어서 떨어지지 않는 경우는 "FALSE"가 뜬다. 물론 다른 숫자들을 입력해서 결과를 확인해 보는 것이 좋다.

위의 명령구를 이용해서 두 수를 입력했을 때 이 두 수가 서로 나누어 떨어지는지의 여부에 따라서 그 결과를 "TRUE"나 "FALSE"로 나타내주는 프로그램(DIVISORP)을 만들 수 있다. 다음 프로그램을 학생들에게 소개하고 이 프로그램이 어떻게 원하는 결과를 OUTPUT 해 주는지 설명해 보도록 한다. 아래와 같은 프로그램을 LOGO PREDICATE라고 부르는데 그 이유는 이 프로그램이 TRUE나 FALSE를 OUTPUT 하기 때문이다. 참고로 OUTPUT 명령어는 PRINT를 사용하는 경우에 주

로 대체 되어서 사용되는데 PRINT 명령어를 사용한 프로그램은 실행결과가 화면에 뜨기 때문에 다른 프로그램에서 이 프로그램을 하위 프로그램으로 사용할 수 없다. 그러나 OUTPUT을 사용할 경우에는 프로그램 실행결과가 화면에 뜨지 않고 내적으로 OUTPUT 하기 때문에 결과를 화면에 띄우려면 프로그램 이름 앞에 PRINT를 쓰면 된다.

두 수를 입력했을 때 이 두 수가 서로 나누어 떨어지는지의 여부에 따라서 그 결과를 "TRUE"나 "FALSE"로 OUTPUT 하는 프로그램(DIVISORP)을 만들기 위해서는 변수가 두 개 필요하고 PRINT 명령어를 OUTPUT으로 바꾸어야 한다. 이제 학생들에게 프로그램을 만들어보게 하고 여러 가지 숫자를 입력해서 프로그램이 원하는대로 만들어졌는지 확인하도록 한다. 예를 들어, 10이 3으로 나누어 떨어지는지 알아보기 위해서는 "PRINT DIVISOR 10 3"을 명령부에 입력하고 "ENTER"키를 누르면 화면에 "FALSE"가 뜬다.

```
완성된 프로그램 :
TO DIVISORP :NUM :DIV
OUTPUT (REMAINDER :NUM :DIV) = 0
END
```

이제는 어떤 수의 모든 약수를 화면에 모두 써주는 프로그램(FACTORS)을 만들고자 한다. 36의 경우를 보면, 1이 약수이면 36도 약수이다 ($1\times36\div1=36$). 그리고 2가 약수이면 $36\div2$도 약수이다. ($2\times36\div2=36$). 이런 식으로 약수들을 나열해보면 다음과 같이 된다.

1, 36
2, 18
3, 12
4, 9
6, 6

9, 4부터는 4, 9와 겹치므로 6 ($=\sqrt{36}$)까지만 확인하면 모든 짝수를 다 쓰게 된다.

이 과정을 프로그램으로 옮기면 다음과 같다.

```
TO FACTORS :NUM :DIV
IF DIVISORP :NUM :DIV [PRINT WORD :DIV '', (:NUM / :DIV))]
IF :DIV > SQRT :NUM [STOP]
FACTORS :NUM :DIV + 1
END
```

위의 프로그램은 학생들이 직접 만들기에는 다소 어려우므로 학생들이 위의 프로그램을 실행할 기회를 제공하고 프로그램이 어떻게 인수를 모두 PRINT 하는지 설명해보도록 하는 것이 적절하다고 생각된다. 위 프로그램의 두 번째 줄에서 WORD :DIV를 WORD '':DIV로 바꾸면 결과에 어떤 변화가 일어나고 왜 그렇게 되는지 알아보도록 할 수 있으며, 이때 프로그램을 실제로 수정해서 알아보게 한다. 위의 프로그램을 간단히 설명하면 다음과 같다. 위의 경우 재귀과정을 사용해서 프로그램을 만들었는데, FACTORS가 끝난 뒤 다시 FACTORS가 시작되는 경우에 :DIV가 :DIV+1로 바뀌어서 숫자가 하나씩 증가하게 되어있고, :NUM이 :DIV로 나누어지는 경우에 [ ] 안이 실행된다. 그러므로 :DIV를 1로 입력하면 1부터 숫자가 하나씩 증가하면서 나누어지는 경우에는 나누는 숫자와 나누어진 숫자를 화면에 PRINT 하게 된다. 위의 36의 경우에서 알 수 있는 것과 같이 $\sqrt{36}$까지만 확인하면 그 다음부터는 반복되므로 나누는 수(:DIV)가 주어진 숫자(:NUM)의 제곱근보다 커지면 프로그램이 종료(STOP)되게 만들어졌다.

이제는 프로그램을 보다 세련되게 만들기 위해서 프로그램 결과에 문장을 추가하고자 한다. 이에 대한 예비과정으로 다음을 시행해 보도록 한다.

```
PRINT [I LOVE MATHEMATICS]

TO NUMBER :NUM
PRINT [MY FAVORITE NUMBER IS] :NUM
END

TO NUMBER :NUM
(PRINT [MY FAVORITE NUMBER IS ] :NUM)
END
```

이제는 학생들에게 FACTOR 36을 명령부에 치면 그 약수들의 짝을 모두 PRINT 하는 프로그램을 만들게 한다. 이때 다음과 같은 문장을 삽입토록 해서 THE PAIR

OF COFACTORS OF "입력한 숫자" ARE: "약수들의 짝"을 PRINT하도록 하는데 이때, 이미 만들어 놓은 하위 프로그램 FACTORS를 사용한다.

이 프로그램을 완성하면 다음과 같다.

```
TO FACTOR :NUM
(PRINT [THE PAIR OF COFACTORS OF] :NUM
[ARE:])
FACTORS :NUM 1
END
```

이 프로그램을 완성한 후 원하는 수를 입력해서 그 수의 약수들을 알아볼 수 있다. 예를 들면 FACTOR 20, FACTOR 66 등을 실행해서 이 수들의 약수가 어떤 것인지 알아볼 수 있다.

(2) 소수 (PRIME NUMBER)

이제는 어떤 숫자를 입력했을때 그 숫자가 소수이면 "TRUE", 소수가 아니면 "FALSE"를 PRINT 하는 프로그램(PRIMEP)을 만들고자 한다. 이 프로그램을 만드는 순서는 설명하면 다음과 같다.

1. :NUM=1이면 "FALSE"이다. (:NUM은 입력하는 숫자)
2. :NUM 이 1 보다 크고 SQRT :NUM보다 작은 약수를 갖으면 "FALSE"이다.(왜냐하면 1과 자신 이외의 약수를 갖으면 소수가 아니다.)
3. 위의 두 조건을 만족하지 않으면 "TRUE"이다.(즉, 1도 아니고 1과 자신 이외의 약수를 갖지 않으므로 소수가 된다.)

PRIMEP 프로그램을 만들기 전에 FACTOR를 만들 때 하위 프로그램 FACTORS를 만든 것과 같이 하위 프로그램(NOFACTORSP)을 먼저 만든다. 다음 프로그램(NOFACTORSP)을 학생들에게 제공하고 이 프로그램을 실행할 경우 어떻게 작동하는지 질문하는 것이 좋다.

```
TO NOFACTORSP :NUM :DIV----------(1)
IF DIVISORP :NUM :DIV [OUTPUT "FALSE]--(2)
IF :DIV > SQRT :NUM [OUTPUT "TRUE]----(3)
NOFACTORSP :NUM :DIV + 1----------(4)
END
```

위의 프로그램(NOFACTORSP)에 6과 7을 실행할 경우에 대해서 설명하면 다음과 같다. (1)에서 NOFACTORSP 6 2를 실행하면 (2)에서 6은 2로 나누어 떨어지므로 "FALSE"인 결과가 나오게 된다. 즉, 6은 소수가 아닌 것이다. 그런데 NOFACTORSP 7 2를 실행하면 (2)에서 7은 2로 나누어 떨어지지 않으므로 (3)으로 넘어가고 2는 $\sqrt{7}$보다 작으므로 (4)로 넘어가고 :DIV가 1 증가해서 다시 (1)로 올라가서 NOFACTORSP 7 3으로 다시 시작하게 되며, (2)에서 7은 3으로 나누어 떨어지지 않으므로 (3)으로 넘어가고 3은 $\sqrt{7}$보다 크므로 "TRUE"인 결과가 나타나게 된다. 즉, 7이 소수라는 결과를 얻게 된다.

이제는 이 프로그램(NOFACTORSP)을 사용해서 어떤 숫자를 입력했을 때 그 숫자가 소수이면 "TRUE", 소수가 아니면 "FALSE"를 PRINT 하는 프로그램(PRIMEP)을 학생들에게 만들어 보도록 한다. NOFACTORSP 프로그램에 소수인지 알아보고자 하는 숫자와 2를 입력하면 소수인지 여부를 알 수 있으므로 알아보고자 하는 숫자가 1인 경우만 생각해 주고 NOFACTORSP를 하위 프로그램으로 사용하면 된다.

완성된 프로그램:

```
TO PRIMEP :NUM
IF :NUM = 1 [OUTPUT "FALSE]
OUTPUT NOFACTORSP :NUM 2
END
```

PRIMEP 프로그램이 완성되고 나면 이를 활용해서 어떤 수가 소수인지 판별해볼 수 있다. 269, 567, 89, 97, 969, 2, 131, 133, 775, 727, 737, 2001, 2010, 2011, …… 또는 1부터 100까지 소수가 몇 개 있는지 알아볼 수도 있다.

(3) 소인수분해

소인수분해 하는 프로그램(PFACTORS)도 다른 프로그램과 비교했을 때 다소 복잡하므로 이를 먼저 학생들에게 제공하고, 실제로 숫자를 대입해서 프로그램이 만

들어진 순서대로 따라갔을 때 어떤 결과가 나올 것인지 예상해보도록 하고, 이를 실제로 실행해서 자신의 결과와 비교해보는 지도방법이 적절하다.

```
TO PFACTORS :NUM :DIV ----------------(1)
IF PRIMEP :NUM [OUTPUT :NUM] ---------(2)
IF DIVISORP :NUM :DIV [OUTPUT SENTENCE
:DIV PFACTORS (:NUM / :DIV) :DIV ----------(3)
PFACTORS :NUM (:DIV + 1) --------------(4)
END
```

위의 프로그램에 PFACTORS 12 2, PFACTORS 15 2를 대입해서 그 결과를 예상하고 이를 실행해서 그 결과와 비교할 수 있다.

PFACTORS 12 2 의 경우, (2)에서 12는 PRIME이 아니므로 다음 단계로 간다. (3)에서 12는 2로 나눌 때 나머지가 없으므로 "2"가 쓰여지고 PFACTOR 12/2가 다시 (1)로 간다. (1)에서 PFACTORS 6 2이므로 (2)에서 6은 PRIME이 아니다. 그러므로 (3)으로 가게된다. (3)에서 6은 2로 나눌 때 나머지가 없으므로 다시 "2"가 쓰여지고, PFACTOR 6/2 2가 다시 (1)로 가서 "3"이 쓰여진다.

이제 PFACTOR를 활용해서 소인수분해 하는 프로그램(PRIMEFACTOR)을 만들 수 있는데 이 부분은 간단하므로 학생들이 만들도록 지도한다.

```
TO PRIMEFACTOR :NUM
PRINT PFACTORS :NUM 2
END
```

위의 PRIMEFACTOR 프로그램을 활용해서 여러 가지 탐구활동을 할 수 있는데, 예를 들면 PRIMEFACTOR 10, PRIMEFACTOR 100, PRIMEFACTOR 1000, PRIMEFACTOR 10000, PRIMEFACTOR 100000, PRIMEFACTOR 1000000 … 등이 갖는 규칙성을 찾을 수 있고, PRIMEFACTOR 20, PRIMEFACTOR 200, PRIMEFACTOR 2000, PRIMEFACTOR 20000, PRIMEFACTOR 200000, PRIMEFACTOR 2000000 … 등이 갖는 규칙성도 알아볼 수 있다.

(4) 최대공약수

최대공약수를 구할 수 있는 프로그램을 만들기 전에 이미 만들었던 FACTOR 프로그램을 활용해서 다음에 나오는 수들의 최대공약수(GCD)를 찾아볼 수 있다.

1) 72, 80 2) 153, 204 3) 2007, 171 4) 24, 36, 108

두 수의 최대공약수를 구하는 EUCLIDEAN ALGORITHM은

1. 큰 수를 작은 수로 나누어서 나머지가 0이면 작은 수가 두 수의 GCD가 된다.
2. 나머지가 0이 아니면 작은 수와 나머지의 GCD가 두 수의 GCD가 된다.

이를 프로그램을 만드는데 그대로 적용해보면 다음과 같다.

```
TO GCD :A :B ----------------------------(1)
IF (REMAINDER :A :B) = 0 [OUTPUT :B] ---(2)
OUTPUT GCD :B (REMAINDER :A :B) ------(3)
END
```

위의 프로그램에 80과 72를 대입하면 어떤 결과가 나오게되는지 학생들이 이를 실행해 보기 전에 프로그램을 보고 예상해 보도록 한다. GCD 80 72의 경우 나머지가 0이 아니므로 (3)으로 간다. (3)에서 GCD 72 8이 되어 (1)로 가서 다시 (2)로 내려오면 72/2는 나머지가 0이므로 "8"을 OUTPUT 한다.

(5) 최소공배수

최소공배수(LCM)를 구할 수 있는 프로그램을 만들기 전에 6과 8의 최소공배수를 찾는 기본적인 방법에 대해서 알아보자. 6의 배수는 6, 12, 18, 24 … 이고 8의 배수는 8, 16, 24, … 이므로 24는 8의 배수이고 6으로 나누어 떨어지는 수이다. 이를 일반적으로 서술하면 다음과 같다.

1. 큰 수를 1부터 순서대로 곱한다.
2. 작은 수가 어떤 곱한 수로 나누어지면 이 곱한 수가 최소공배수(LCM)가 된다.

위의 ALORITHM을 그대로 프로그램으로 만들면 다음과 같이 된다.

```
TO LCM2 :A :B :N ------------------------(1)
```

```
IF DIVISORP (:N * :A) :B [OUTPUT :N * :A]---(2)
LCM2 :A :B :N + 1 ----------------------(3)
END
```

위의 프로그램을 학생들에게 제공하고 어떤 식으로 프로그램이 작동하는지 질문해 보고 두수를 입력했을 때 최소공배수를 OUTPUT 하는 프로그램을 만들도록 유도할 수 있다. 이때 이미 만들었던 PRIMEFACTOR의 경우를 참고하면 쉽게 만들 수 있다.

```
TO LCM :A :B
OUTPUT LCM2 :A :B :1
END
```

이제는 위의 프로그램 LCM에 8과 6을 입력하면 프로그램이 어떻게 작동하는지 이를 학생들에게 설명하도록 유도한다.

이를 설명하면 다음과 같다. LCM 8 6을 실행시키면, (1)에서 LCM2 8 6 1이 되고, (2)에서 6×1은 8로 나누었을 때 나머지가 0이 아니므로 (3)에서 LCM2 8 6 2가 된다. 그러므로 다시 (1)을 지나서 (2)에서 6×2는 8로 나누었을 때 나머지가 0이 아니므로 (3)에서 LCM2 8 6 3이 된다. (1)을 거쳐서 (2)에서 6×3은 8로 나누었을 때 나머지가 0이 아니므로 (1)을 지나서 (2)에서 6×4는 8로 나누었을 때 0이므로 24를 OUTPUT 하게 된다.

이제는 위의 프로그램을 이용해서 A*B=GCD*LCM이라는 관계를 발견하도록 유도할 수 있다. A와B를 주고 A*B, GCD, LCM, GCD*LCM을 계산하도록 해서 그 결과들을 표에 입력하도록 해서 관계를 확인하게 할 수 있다.

(6) COUNTER

COUNTER 프로그램을 만들기 전에 학생들에게 이 프로그램을 실행했을 경우 화면에 나타나게 될 상황을 설명하고 재귀과정을 사용해서 프로그램을 만들어보도록 한다. 프로그램을 만든 후 명령부에 COUNTER 1을 입력하고 ENTER키를 누르면 화면에 1, 2, 3, 4, 5, ... 와 같이 숫자들이 나타나야 한다.

```
완성된 프로그램:
TO COUNTER :N
PRINT :N
COUNTER :N + 1
END
```

위의 절차를 실행하면 컴퓨터 메모리 용량에 따라서 어떤 수에 도달하면 멈추게된다. 이 프로그램(COUNTER)을 원하는 수에서 멈추도록 만들려면 조건문을 삽입해야 한다. 이 조건문은 이미 위에서 소개했던 IF :N > 100 [STOP] 인데 이를 (1), (2), (3) 중 어느 부분에 넣는 것이 좋은지 묻고, 그 이유를 설명하도록 하는 것은 학생들이 재귀과정을 이해하는데 도움이 된다.

```
TO COUNTER :N ------------------------(1)
PRINT :N-----------------------------(2)
COUNTER :N + 1-----------------------(3)
END

완성된 프로그램:
TO COUNTER :N
IF :N > 100 [STOP]
PRINT :N
COUNTER :N + 1
END
```

COUNTER는 수를 1부터 위로 세는 프로그램이다. 이번에는 프로그램 COUNTER를 참고로 해서 수를 거꾸로 세는 프로그램(COUNTDOWN)을 학생들에게 만들어 보도록 할 수 있다. 이때 1에서 숫자가 멈추기 위한 조건문의 위치에 대해서 생각해 보도록 한다.

```
완성된 프로그램:
TO COUNTDOWN :N
IF :N < 1 [STOP]
PRINT :N
COUNTDOWN :N - 1
END
```

위의 COUNTER는 100에서 멈추게 되어 있다. 이를

50번째에 멈추게 하려면 100을 50으로 수정해야 한다. 이때 원하는 숫자까지 셀 수 있는 프로그램을 어떻게 만들 수 있는지 생각하도록 한다. 이 프로그램을 만들기 전에 LOGO에서는 변수를 여러 개 사용할 수 있다는 것을 상기시킨다.

완성된 프로그램:

```
TO COUNTER :N :LIMIT
IF :N > :LIMIT [STOP]
PRINT :N
COUNTER :N + 1 :LIMIT
END
```

(7) N항까지의 합 (SUMN)

이번에는 COUNTER가 만든 수열의 N항까지의 합을 계산하는 프로그램을 만들려고 한다. 이 프로그램을 만들기 위해서 귀납적인 접근방법을 사용할 수 있다. 첫째로, 1항까지의 합은 1이다. 둘째로, N항까지의 합은 N-1항까지의 합에 N을 더한 값이 된다. 이 두 가지 내용을 LOGO를 사용해서 그대로 옮기면 N항까지의 합을 구하는 프로그램이 만들어진다.

```
TO SUMN :N
IF :N = 1 [PRINT 1 STOP]
PRINT :N + SUMN :N - 1
END
```

그런데 위의 절차를 실행해보면 "SUMN did't report anything to + in SUMN"이라는 에러메시지가 뜬다. 이러한 메시지가 화면에 뜨는 이유는 "PRINT" 명령어 때문이다. SUMN이 자신을 부르는 과정에서 발생하는 에러로, PRINT 명령어가 나오면 PRINT 다음의 것을 화면에 쓰고 멈추게 된다. 그러므로 이를 불러서 사용할 수 가없다. 이 에러를 없애기 위해서는 PRINT를 OUPUT으로 바꾸면 된다.

완성된 프로그램:

```
TO SUMN :N
IF :N = 1 [OUTPUT 1 STOP]
OUTPUT :N + SUMN :N - 1
END
```

위의 프로그램은 재귀과정을 이용해서 만든 것인데 위와 동일한 결과를 나타내는 프로그램을 1+2+3+4+…+N=N(N+1)/2이라는 식을 사용해서 만들 수도 있다. 이 식을 프로그램으로 옮기면 아래와 같다.

완성된 프로그램:

```
TO F :N
OUTPUT  :N * (:N + 1) / 2
END
```

위의 두 가지 프로그램이 같은 결과를 나타내는지 같은 수를 입력해서 확인해 보도록 유도한다. 예를 들면, SUMN 100과 F 100의 결과를 비교하는 것이다.

(8) 등차수열

등차수열을 만드는 프로그램은 위에서 이미 만들었던 COUNTER를 참고로 하면 비교적 용이하게 만들 수 있다. 등차수열을 만들기 위해서는 초항과 공차를 알아야 하므로 이 프로그램을 만들 때 변수는 두 개를 사용해야 한다. 이 프로그램의 이름을 ARITHSEQ이라고 하고 초항을 A, 공차를 D라고 하면 대략 다음과 같은 모양의 프로그램을 만들 수 있다.

```
TO ARITHSEQ :A :D
PRINT :A
ARITHSEQ ·····
END
```

위와 같이 프로그램의 개형을 주고 학생들에게 프로그램을 완성하도록 충분한 시간을 허락한다. 위의 프로그램에 실제로 두 수를 대입해서 생각해 보도록 유도하고, ARITHSEQ는 변수를 두 개 취한다는 사실을 주지시킨다. 또한 위의 프로그램에서 PRINT 하는 변수는 :A이므로 ARITHSEQ 다음에 나오는 두 변수 중에서 첫번째 변수를 조작해서 등차수열을 만들어야함을 학생들에게 주지시킨다.

완성된 프로그램:

```
TO ARITHSEQ :A :D
PRINT :A
ARITHSEQ (:A + :D) :D
END
```

그런데 위의 프로그램(ARITHSEQ)을 실행시키면 화면에서 순식간에 수열을 PRINT하기 때문에 처음의 몇 항을 확인하기가 어렵다. 그러므로 원하는 항까지만 수열을 만들려고 하면 위의 프로그램에 변수를 하나 더 추가해서 원하는 항까지 수열을 만들 수 있다. 이때 세 번째 변수는 항을 세는 역할을 해야 하고 프로그램이 한번 실행될 때마다 수가 하나씩 줄어야 한다. 학생들에게 이미 만들었던 COUNTDOWN을 참고로 해서 위의 프로그램을 완성하도록 하면 된다.

완성된 프로그램:

```
TO ARITHSEQ :A :D :N
IF :N = 0 [STOP]
PRINT :A
ARITHSEQ (:A + :D) :D (:N - 1)
END
```

이번에는 원하는 수열을 전부 쓰는 것이 아니고 원하는 항만을 나타내는 프로그램을 만들고자 한다. 즉, ARITHSEQ 2 3 4를 실행시키면 2, 5, 8, 11이 화면에 나타나게 되는데 이때 새로 만들고자하는 프로그램은 NTHARITH 2 3 4를 실행시키면 화면에 11만 뜨도록 프로그램을 만드는 것이다. 이러한 프로그램을 만들려면 결과를 PRINT 하기 전에 혼자서 재귀과정을 거친 후 원하는 항이 되면 그 항만 PRINT 하게 프로그램을 만들면 된다. 이 프로그램도 위의 ARITHSEQ를 참고로 해서 만들 수 있다.

완성된 프로그램:

```
TO NTHARITH :A :D :N
IF :N = 1 [PRINT :A STOP]
NTHARITH (:A + :D) :D (:N + 1)
END
```

NTHARITH는 재귀과정으로 등차수열의 원하는 항을 구하는 프로그램이었다. 자연수의 N항까지의 합을 구할 때 1+2+3+……+N=N(N+1)/2를 사용한 것과 같이 등차수열의 N번째 항을 구할 때 :A + (:N - 1) * :D를 사용해서 다른 프로그램(F2)을 만들어서 원하는 항을 구할 수 있다. 이 프로그램도 프로그램 "F"를 참고로 해서 학생들이 만들어보도록 할 수 있다.

완성된 프로그램:

```
TO F2 :A :D :N
OUTPUT :A + (:N - 1) * :D
END
```

NTHARITH와 F2를 비교해보기 위해서 NTHARITH 3 2 10 과 F2 3 2 10을 실행해서 같은지 비교해 볼 수 있다.

이제는 N항을 나타내는 NTERM이라는 하위 프로그램을 이용해서 N항까지의 합을 계산해주는 프로로그램을 만들고자 한다. NTERM은 위의 F2를 이용해서 다음과 같이 간단히 나타낼 수 있다.

```
TO NTERM :A :D :N
OUTPUT F2 :A :D :N
END
```

N항까지의 합은 N항 + (N - 1)항까지의 합으로 나타낼 수 있으므로 N항을 나타내는 NTERM에 (N - 1)항까지의 합을 나타내는 SUMARITH를 더하면 되며, 이를 그대로 프로그램으로 옮겨 놓으면 된다.

```
TO SUMARITH :A :D :N
IF :N = 1 [OUTPUT :A]
OUTPUT (NTERM :A :D :N) + SUMARITH :A :D
(:N - 1)
END
```

이 프로그램을 가지고 여러 가지 등차수열을 대입해서 N항까지의 합이 맞는지 확인해 볼 수 있다. 예를 들어, SUMARJTH 2 3 5를 실행하면 초항이 2이고 공차가 3인 등차수열의 5항까지의 합을 구할 수 있다.

(9) 등비수열

등차수열을 만들고 나면 등비수열을 만드는 프로그램(GEOMSEQ)을 만드는 것은 수월해진다. 초항과 공비가 주어졌을 때, 등비수열을 만드는 프로그램은 등차수열을 만드는 프로그램을 참고로 하여 만들도록 한다.

```
완성된 프로그램:
TO GEOMSEQ :A :R
PRINT :A
GEOMSEQ (:A * :R) :R
END
```

원하는 항까지만 등비수열을 나타내고 싶으면 등차수열의 경우와 마찬가지로 변수를 하나 추가해서 만들면 된다. 이 경우에도 마찬가지로 학생들에게 등차수열 만드는 프로그램을 참고하도록 한다.

```
완성된 프로그램:
TO GEOMSEQ :A :R :N
IF :N = 0 [STOP]
PRINT :A
GEOMSEQ (:A * :R) :R :N - 1
END
```

등비수열에서 N번째 항만을 나타내는 프로그램을 만들기 위해서는 등차수열에서 만든 프로그램 NTHARITH를 참고하게 하면 쉽게 만들 수 있다.

```
완성된 프로그램:
TO NTHGEOM :A :R :N
IF :N = 1 [PRINT :A STOP]
NTHGEOM (:A * :R) :R (:N - 1)
END
```

이제는 등비수열의 N번째 항을 OUTPUT 하는 프로그램을 만드는데 $An=ar^{n-1}$을 사용하고자 한다. 그런데 이 식에서는 멱이 나타나므로 멱을 나타낼 수 있는 하위 프로그램을 만들어야 한다. 이 프로그램을 POWER라고 부르기로 하자. POWER 프로그램을 만들기 위해서는 다음 세 가지 성질을 알고 있어야 한다.

- 음수멱은 다음과 같이 나타낼 수 있다. $N^{-P}=1/N^{P}$
- 0이 아닌 수의 0제곱은 1이다.
- 모든 양의 정수에 대해서 $N^{P} = N * N^{P-1}$

위의 세 가지 성질을 그대로 옮기면 멱(POWER)을 만드는 프로그램이 된다.

```
완성된 프로그램:
TO POWER :N :P
IF :P < 0 [OUTPUT 1 / POWER :N (-1 * :P)]
IF :P = 0 [OUTPUT 1]
OUTPUT :N * POWER :N :P - 1
END
```

이제 위의 POWER를 사용하고 등차수열의 F2를 참고로 하면 N번째 항을 OUTPUT하는 프로그램을 만들 수 있다.

```
완성된 프로그램:
TO NGEOM :A :R :N
IF :N = 1 [OUTPUT :A STOP]
OUTPUT (A: * POWER :R (:N - 1))
END
```

등비수열의 N항까지의 합을 구하는 프로그램은 등차수열의 합을 구하는 프로그램인 SUMARITH를 참고로 하면 비교적 쉽게 프로그램을 만들 수 있다. 등비수열에서도 N항까지의 합은 N항에 N-1항까지의 합을 더하면 되므로 NGEOM에 SUMGEO를 더하면 된다.

```
완성된 프로그램:
TO SUMGEO :A :R :N
IF :N = 1 [OUTPUT :A STOP]
OUTPUT (NGEOM :A :R :N) + (SUMGEO :A :R
(:N - 1))
END
```

등비수열의 경우도 이미 만들어 놓은 프로그램을 활용해서 여러 가지 등비수열을 만들어보고, N항에 대해서 알아보고, N항까지의 합을 직접 계산해 볼 수 있다.

NGEOM 2 3 5 는 초항이 2이고 공비가 3인 등비수열의 5번째 항이 어떤 수인지 알아볼 수 있고, SUMGEO 2 3 5는 초항이 2이고 공비가 3인 등비수열의 5번째 항까지의 합을 계산해 준다.

(10) Fibonacci 수열

Fibonacci 수열은 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 …… 과 같이 나타낼 수 있다. Fibonacci 수열의 초항과 둘째항은 1이고, 셋째항부터는 이전에 나오는 2항의 합이다. 이러한 사실을 그대로 적용하면 Fibonacci 수열을 만드는 프로그램을 만들 수 있다. 프로그램에 두 항의 합을 나타내야하므로 항을 나타내는 변수가 두 개 필요하다. 이를 TERM1, TERM2 라고 주고 학생들이 프로그램을 만들도록 유도한다. 그러면 아래와 비슷한 프로그램을 만들어서 Fibonacci 수열을 만들려고 하는 학생이 있을 것이다.

```
TO FIB2 :TERM1 :TERM2
PRINT :TERM1 + :TERM2
FIB2 :TERM1 :TERM2
END

TO FIB3 :TERM1 :TERM2
PRINT :TERM1
FIB3 (:TERM1 + :TERM2) :TERM2
END
```

F2와 F3를 직접 실행해보기 전에 TERM1과 TERM2에 각각 1을 대입해서 어떤 숫자를 PRINT 하게 되는지 알아보게 한다. 그러면 이 두 프로그램은 원하는 Fbonacci 수열을 만들지 않는다는 것을 알게된다. F3의 경우는 Fbonacci 수열을 만들 것처럼 보이지만 실제로 해보면 양의 정수를 만들게 된다. 이 경우에 TERM2는 계속 1이라는 값만을 갖게 되므로 이점을 착안하면 프로그램을 바르게 고칠 수 있다. 문제는 TERM2가 1 이라는 값을 가지고 계속 공전하는 것이므로 "FIB3 (:TERM1 + :TERM2) :TERM2"를 "FIB3 :TERM2 (:TERM1 + :TERM2)"로 바꾸면 된다.

완성된 프로그램:

```
TO FIB :TERM1 :TERM2
PRINT :TERM1
FIB :TERM2 (:TERM1 + :TERM2)
END
```

위의 프로그램을 실행시키면 화면 위에 Fibonacci 수열이 너무 빠르게 PRINT 되므로 원하는 항까지만 수열을 만들기 위해서는 등차수열과 등비수열의 경우처럼 변수를 하나 더 만들면 된다. 학생들에게 등차수열과 등비수열을 참고로 해서 원하는 항까지 Fibonacci 수열을 만들도록 한다.

완성된 프로그램:

```
TO FIB :TERM1 :TERM2 :N
IF :N = 0 [STOP]
PRINT :TERM1
FIB :TERM2 (:TERM1 + :TERM2) (:N - 1)
END
```

Fibonacci 수열의 두 항의 비는 황금비에 수렴한다. 이 사실을 LOGO 프로그램으로 확인해 볼 수 있는데 다음과 같은 프로그램을 제공하거나 관심 있는 학생에게 직접 만들게 해서 직접 황금비에 수렴하는 사실을 알아볼 수 있다.

완성된 프로그램:

```
TO FIBRATIO :TERM1 :TERM2 :N
IF :N = 0 [STOP]
PRINT :TERM2 / :TERM1
FIBORATIO :TERM2 (:TERM1 + :TERM2) (:N - 1)
END
```

이때 FIBORATIO 1 1 5, FIBORATIO 1 1 10, FIBORATIO 1 1 20 등을 실행시키면서 과연 두 항의 비가 수렴하는지 알아볼 수 있다.

Fibonacci 수열에서도 등차수열과 등비수열처럼 일반항(N항)을 구하는 프로그램을 만들어보고자 한다. 초항과 둘째항은 1이고, 셋째항부터는 이전의 두 항을 더해서 만들 수 있으므로, 이 두 가지 사실을 그대로 프로그

램으로 옮기면 된다. 이때 초항과 둘째항은 모두 1이므로 LOGO에서 사용하는 명령어 "OR"의 사용법을 설명하고 프로그램을 만들도록 한다. IF OR (:N = 1) (:N = 2) [OUTPUT 1]은 N이 1 또는 2인 경우 1을 OUTPUT하게 된다. 이 경우 OUTPUT 대신에 PRINT 명령어를 사용하면 나중에 다른 프로그램에서 FIBN을 하위 프로그램으로 사용할 수 없게 된다.

```
완성된 프로그램:
TO FIBN :N
IF OR (:N = 1) (:N = 2) [OUTPUT 1]
OUTPUT (FIBN :N - 1) + (FIBN :N - 2)
END
```

Fibonacci 수열에서도 N항까지의 합은 N항에 N-1항까지의 합을 더하는 것이므로 등차수열과 등비수열의 경우와 비교해 보면 쉽게 만들 수 있다.

```
TO SUMFIBN :N
IF :N = 1 [OUTPUT 1]
OUTPUT (FIBN :N) + SUMFIBN (:N - 1)
END
```

IV. 요약 및 적용

LOGO 언어는 Piaget의 반영적 추상화 개념과 계산적 이론 및 인공지능의 결합으로 학생들의 수학적인 사고력을 신장시키기 위해서 만들어진 컴퓨터 언어로, 수학교육 선진국에서는 수학교육에서 컴퓨터를 활용하기 시작하면서부터 많은 교사 양성과정에서 이를 가르친 바 있다. LOGO 언어는 학교 현장에서 관심 있는 교사들에 의해서 학생들에게 소개되었으며, 이미 서론에서 언급한 바와 같이 그 동안 이에 관한 많은 연구가 진행되어 왔다. 그 동안의 LOGO를 사용해서 가르친 수준은 주로 초등학교 수준으로 중등학교에서 LOGO의 활용에는 구체적인 아이디어가 제공된 경우가 많지 않았다. 이러한 이유로, 본 연구는 LOGO의 중등수학교육의 활용방안을 제시했는데 여기에서 제시한 내용들은 참고 가능한 몇 가지 활용 가능한 예를 제시한 것으로, 이를 실제로 우리 나라의 학교현장에서 활용하려면 이를 활용하려는 교사는 가르치는 학생들의 학년, 수준, 사용 가능 시간 등을 고려해서 활용 부분과 방법을 재구상 해야 할 것으로 생각된다. 특히, 특별활동으로 수학반에서 LOGO를 가르치면 더욱 자세히 공부할 수도 있다. 이를 정규 수학 수업시간에 사용하고자 한다면, 이미 만들어진 프로그램을 학생들에게 시연하면서 보여줄 수도 있다.

본 연구에서 제한된 주제를 소개하였는데, 이러한 프로그래밍 활동을 통해서 학생들이 수학적 사고능력을 어느 정도 신장할 수 있는지, 또한 수학에 대한 태도에는 어느 정도 영향을 미치는지를 알기 위해서는 위에 제시한 예들을 기초로 프로그래밍 활동을 실제로 학생들에게 실행해 보는 후속 연구가 필요할 것으로 예상된다.

참 고 문 헌

김수환 · 이재학 (1992). 논리적 사고력 신장을 위한 LOGO프로그래밍 활동의 효과분석, 한국수학교육학회지 <수학교육> **31(1)**, pp.11-22.

류희찬 · 이지요 (1993). 수학교육에서의 시각화의 중요성과 LOGO, 대한수학교육학회 논문집 **3(1)**, pp.75-85.

류희찬 (1994). LOGO를 통한 수학학습: 그 내용과 방법, 한국수학교육학회지 <수학교육> **33(2)**, pp. 197-207.

백영균 · 류희찬 (1990). 「로고 아동과 컴퓨터」, 서울: 양서원.

Denenberg, S. (1993). Using Logo to Develop Problem Solving Skills, *Journal of Computer Science Education* **7(3)**, pp.29-35 .

Grandgenett, N. (1990). Using LOGO Programming to Teach Analogical Reasoning in Science and Mathematics Education, *Journal of Computers in Mathematics and Science Teaching* **10(3)**, pp.29-36.

Jensen, R.J. & williams, B.S. (1993). Technology: Implications for Middle Grades Mathematics, In D. T. Owens (Ed.) *Research Ideas for The Classroom Middle Grades Mathematics*(pp.225-243), NCTM: Research Interpretation Project.

Kern, J. & Mauk, C. (1990). Exploring Fractals - A Problem-Solving Adventure Using Mathematics and Logo, *Mathematics Teacher* **83(3)**, pp.179-85.

Lemerise, T. (1990). Can We Integrate LOGO into the Regular Mathematics Curriculum and Still Preserve the LOGO Spirit? *For the Learning of Mathematics* **10(3)**, pp.17-19.

Maddux, C. (1992). Logo: The Case for a Cautious Advocacy, *Computers in the Schools* **9(1)**, pp.59-79.

McAllister, A. (1991) An Analysis of Problem Solving Strategies in LOGO Programming Using Partially Automated Techniques, Paper presented at the Annual Meeting of the American Educational Research Association, Chicago, Illinois, April 3-7.

Nevile, L. (1992). Logo and Mathematics Education LME5, *Proceedings of the Annual Conference* (5th, Lake Yinaroo, Queensland, Australia, April 1-5.

Olive, J. (1991). Logo Programming and Geometric Understanding: An In-Depth Study, *Journal for Research in Mathematics Education* **22(2)**, pp.90-111.

Papert (1980). *Mindstorms: Children, computers, and powerful ideas*, New York: Basic Books.

Schuyten, G. & Valcke, M., Ed. (1990). Teacher Education in Logo-Based Environments. Commission of the European Communities, Brussels., Belgium.

Swan, K. & Black, J. (1990). Results of Four Studies on Logo Programming, Problem Solving, and Knowledge Based Instructional Design, Paper presented at the international Conference on Technology and Education, Brussell, Belgium, March 20-22.

Taylor, R. (1980). *The Computer in The School: Tutor, tool, tutee*, New York: Teachers College Press.

Yusuf, M. (1990). LOGO Based Instruction, Paper presented at the Annual Meeting of the Midwestern Educational Research Association, Chicago, Illinois. October 20.

A Study of LOGO in Secondary School Mathematics Classroom

Whang, Woo-Hyung
Department of Mathematics Education, College of Education, Korea University,
Seoul, 136-701, Korea; e-mail: wwhang@kuccnx.korea.ac.kr

The purpose of the study was to suggest a few fundamental ideas to secondary mathematics teachers regarding using LOGO in their classrooms. Even though this software is "classic" in mathematics lessons in the States and western countries, Korean secondary mathematics teachers did not have much opportunities to become familiar with this software and its applications in their classrooms.

This article offered a few suggestions with the specific guidelines. Basic commands were also listed for those who are not familiar with this software. Since the current Korean educational curriculum do not allow to use LOGO in regular classrooms as substitute lessons, it is recommended to implement these ideas in extra curriculum lessons or "Math Club" as trial basis.