韓國컴퓨터情報學會 論文誌
第16卷 第12號, 2011. 12.

2011-16-12-3-2

중등 정보과학 영재 사사지도 행렬중심 교수학습 자료 개발

이 형 봉*

Development of a Matrix-focused Instructional Materials for Personal Education for the Gifted Middle School Students of Computer Science

Hyung Bong Lee*

요 약

최근 몇 년 사이, 학부모는 물론 국가 차원에서 영재 교육에 대한 관심이 높아짐에 따라 영재 교육에 관한 많은 연구들이 발표되고 있다. 그러나, 그들 대부분의 초점이 개괄적이고 총론적인 관점에서 영재 교육 프로그램의 구축, 운영, 관리, 평가 등에 있고, 정작 교육 프로그램의 성패와 직결되는 구체적인 교수학습 주제를 다루는 연구는 극히 드물다. 오늘날 영재 교육의 필요성이나 효율성이 재론되고 있는 원인 중의 하나는 바로 영재 교육의 특수성들이 고려된 영역별 훌륭한 교육 시나리오의 빈곤에 있음을 부인할 수 없다. 따라서, 이 연구에서는 중등 정보과학 영재 사사 지도과정에서 적용되었던 탐구 내용을 소개함으로써 정보분야 영재교육을 위한 교수학습 자료 개발에 작은 보탬이 되고자한다. '컴퓨터과학에서 행렬의 필요성과 활용'이란 주제의 이 교수학습 자료는 중등 수학의 1차 연립방정식 풀이 과정을 기반으로 창의적 문제 발견 및 해결을 유도하도록 구성되었고, 수학과 컴퓨터 과학과의 긴밀한 연계성 및 선형대수학의 기초 개념 이해에도 유익하다.

▸ Keyword : 중등 컴퓨터 과학 영재, 교수학습 자료, 행렬, 1차 연립방정식

Abstract

In recent years, parents of students and government have been taking a growing interest in education for the gifted students and there are many research reports about the gifted education. Most of the reports, however, focuses on the conceptional feature of the gifted education program

• 제1저자 : 이형봉 • 교신저자 : 이형봉
• 투고일 : 2011. 08. 24, 심사일 : 2011. 09. 27, 게재확정일 : 2011. 10. 08.
* 강릉원주대학교 컴퓨터공학과(Dept. of Computer Science & Engineering, Gangneung-Wonju National University)
※이 연구는 강릉원주대학교 과학영재교육원의 지원을 받았음

such as organization, operation, management, evaluation, etc,. In other words, there are very few researches on instructional materials for gifted students even though the materials is a critical factor for successful education programs. So, this paper introduces a lecture notes used in a personal education for gifted students to contribute in developing education contents in computer science area. The instructional materials titled as "The Necessity and Application of Matrix in Computer Science" is based on linear equation to usher the students into creative problem recognition and groping for solutions. Also, the instructional materials is useful for students to understand the tight mathematics-computer science relationship and the basic concept of liner algebra.

▸ Keyword : Gifted Student of Computer Science, Instructional Materials, Matrix, Linear Equation

I. 서 론

천재 (天才, genius)는 보통 사람에 비해 선천적으로 뛰어난 정신 능력을 가지고 있는 사람으로, 어린 시절부터 재능을 보이며 신동이라 불리는 경우가 많고, 한 영역에서 대가가 되거나, 미지의 영역을 개척하면서 위대한 업적을 남긴다[1]. 이와 같이 천재에 대한 정의는 비교적 명확한데, 영재에 대한 정의는 무엇일까? 영재에 대한 정의는 학자, 국가, 기관에 따라 매우 다양한데, 우리나라의 영재교육진흥법[2]은 "재능이 뛰어난 사람으로 타고난 잠재력을 계발하기 위하여 특별한 교육을 필요로 하는 자"와 같이 정의하고 있다. 즉, 영재는 특정 분야에서의 뛰어난 잠재 능력의 소유자이기는 하지만, 그 잠재력의 충분한 발휘를 위해서는 특별한 교육이 필요하다는 것이다. 이러한 영재의 정의는 그림 1에 보인 교육학자 렌줄리의 세 고리 개념[3]과도 상통한다. 즉, 렌줄리는 높은 수준의 지적 능력 대신, 새로운 것에 대한 호기심(창의성)과 과제에 대한 집중력을 영재성의 요소로 강조하였다. 앞에서 제시된 영재성 요소들은 정보의 처리를 주목적으로 하는 컴퓨터과학 분야에서 그 적합성이 더욱 높은 것으로 판단된다. 일선 대학의 전산 관련 전공 담당 교수들의 주요 의견은 '요즘 학생들은 논리적인 생각을 해야 하는 프로그래밍을 싫어하고, 조금 복잡한 문제에 접하면 포기해버린다. 전산학을 전공하기 위해서는 우선 인내와 끈기의 성품이 필요하다'라는 것으로 과제집착력을 호소하고 있다. 또한 크누스[4]는 프로그래밍을 예술에 비유하여 컴퓨터과학에서 창의성의 필요성을 암시하고 있는데, 사실 정보처리 분량이 커질수록 이를 효과적으로 해결하기 위한 알고리즘 및 프로그램의 구현을 위해서는 고도의 창의성이 요구됨은 두말할 나위가 없다.

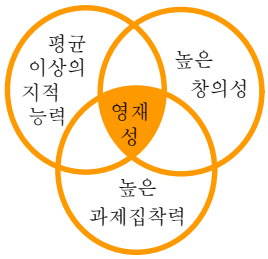

그림 1. 렌줄리의 세 고리 개념
Fig. 1. The Concept of Renzulli's Three-Ring

정보과학 혹은 컴퓨터과학 영재의 정의는 "첨단 정보기기의 활용 능력이 우수하며, 정보기기의 활용을 즐기는 자로서, 자신의 정보과학적인 지적 능력과 정의적 특성을 최대한 발휘하여 첨단 정보과학이론을 정립하고, 정보과학적인 시스템을 설계 및 구현할 수 있으며, 유용한 정보와 지식을 지속적으로 창출할 수 있는 자"[5] 등 학자에 따라 다양하고 모호하나, 대체로 "정보를 처리하는 과정에서 도구 활용 능력과 문제 해결을 위한 알고리즘 창출 능력이 뛰어난 자"정도로 정리할 수 있다. 이 정의에 입각하여 정보과학 영재의 교육은 크게 도구 활용 분야와 문제해결(프로그래밍) 분야로 분류될 수 있다. 대부분의 영재교육이 단순히 기존 지식에 대한 이해과정, 즉 지식 확대 과정으로 이루어지고, 창조적인 문제 발견과 문제 해결 능력을 발굴하는 교육이 미흡하다는 지적이 있는데, 이는 곧 정보과학에서 도구를 이용한 컴퓨터의 활용이나 정보시스템 자체에 관한 지식 확대 목적에 치우쳐 있다는 내용으로 해석할 수 있다. 이런 현상의 가장 큰 원인은 영재들의 능력을 배양할 효과적인 교수학습 자료의 부재에 있고, 이로 인해 각급 영재교육이 검증되지 않은 폐쇄적 교육과정으로 운영되고 있다[6,7]. 따라서, 이 연구에서는 중등 정보과학 영재 사사지도 과정에서 적용되었던 교육 자료를 하나의 개방된 교

수학습 자료로 제안하여, 정보과학 영재교육의 효율성 제고에 기여하고자 한다. “컴퓨터과학에서 행렬의 필요성과 활용”이란 주제의 이 교수학습 자료는 중등 수학의 1차 연립방정식 풀이 과정을 기반으로 창의적 문제 발견 및 해결을 유도하도록 구성되었다. 또한, 이 학습 자료는 고등학생 및 대학 신입생들이 수학과 컴퓨터과학과의 긴밀한 연계성 및 선형대수학의 기초 개념을 이해하거나, 게임, 인터넷, 워드 등 컴퓨터의 단순한 활용의 고정 관념을 넘어 컴퓨터의 숨은 가치를 새롭게 인식하는 데에도 도움이 될 수 있다.

이 논문의 Ⅱ 장에서는 국내 정보과학 영재 교육과정에 대해 간단히 고찰하고, Ⅲ 장에서는 제안된 교수학습 자료의 내용을 중등 영재의 관점 즉, 중등 영재 독자를 목표로 기술한다. Ⅳ 장에서는 이 학습 자료의 적용사례를 통한 효과성을 평가하고, 마지막 Ⅴ장에서 결론으로 이 논문을 맺는다.

Ⅱ. 관련 연구

1. 영재 교육기관 및 교육과정

1.1 영재 교육기관

우리나라의 영재교육은 2003년부터 시작되어 현재 영재학교, 영재학급, 영재교육원, 영재학원 등 크게 네 종류의 교육기관에서 이루어지고 있다. 영재 학교는 국제중·고교, 과학고, 특수 목적고 등과 같이 정규 학교 형태로 운영되고, 영재학급은 지방 교육청에 의해서 운영되며, 영재교육원은 대학 부설로 운영된다. 영재 학원은 지자체 등 공공기관의 인증을 받아 학원 형태로 운영된다. 600개 이상의 각급 영재 교육기관이 존재하는 경기도의 경우[8]에서 보는 바와 같이 현재 국내 영재 교육기관은 수 없이 많지만, 이 논문에서는 전국 25개 대학 부설 영재교육원을 지칭하기로 한다.

1.2 영재 교육과정

국내 영재 교육기관은 주로 수학과 기초과학(물리, 화학, 생물, 지구과학)을 위주로 하여, 대개 초등심화(5-6학년, 년 70시간 이상), 중등심화(1~2학년, 년 100시간 이상), 중등사사(3학년, 년 40시간 이상)과정을 운영하는데, 초등 및 중등심화는 20여명의 반으로 구성되고, 중등사사는 특정 탐구 주제 하에 지도 교수(교사)가 2~3명을 대상으로 진행한다. 정보과학 분야는 교육 기관에 따라 운영 여부가 일정하지 않다.

2. 정보영재 교육과정 및 교수학습 자료

2.1 정보영재 교육과정

[5]는 한국교육개발원, 카이스트 IT영재교육원, 아주대학교 과학영재교육원, 서울교육대학교 과학영재교육원, 경북대학교 정보영재교육원 등을 대상으로 정보영재 분야 교육과정을 분석하였는데, 그 특징 및 문제점, 그리고 개선안을 표 1~3에 각각 요약하였다. 이 결과에서 주목되는 부분은 대부분이 정보영재 교육이 기계적 프로그래밍 분야에 편중되어 있고, 영재들의 잠재력 계발에 주안을 둔 교수학습 자료의 개발이 필요하다는 사실이다.

표 1. 정보영재 분야 교육과정 특징
Table 1. Characteristics of the Education Curriculum for the Gifted Student in Computer Science

• 자체 개발한 교육과정에 의거 교육을 실시하고, 영재교육원 사이의 교류는 없음 • 프로그래밍교육을 위주로 하며, 그 외 분야는 다양함(사고력 신장, 기초소양, 올림피아드 목표 기초 소양) • 교육 유형은 주말교육, 방학 중 집중교육(합숙), 원격 교육

표 2. 정보영재 분야 교육과정의 문제점
Table 2. Problems of the Education Curriculum for the Gifted Student in Computer Science

• 교육과정이 상이하고 교류가 없으므로 교육 기관별 포괄 영역이 협소 • 창의력 신장을 위한 교육보다는 기계적 프로그래밍 교육에 치중 • 지적 우수성만을 추구하는 수학·과학 분야에 편중(정보 분야 영재는 수학·과학의 절반에도 미치지 못함)

표 3. 정보영재 분야 교육과정의 개선안
Table 3. How to Resolve the Problems of the Education Curriculum for the Gifted Students of Computer Science

• 국가 수준의 정보 영재 교육과정의 마련 및 영재교육원들 사이의 정기적인 교류 • 창조적 사고력을 배양시킬 수 있는 창의성 신장 프로그램의 강조 • 어플리케이션 기초소양 능력에 대한 지나친 의존 지양 • 영재들의 잠재력 계발에 주안을 둔 교수학습 자료의 개발

2.2 정보과학 영재 교수학습 자료

정보과학 영재를 위한 교수학습 자료 또한 교육과정만큼이나 폐쇄적이어서 교육기관별로 자체 제작되어 사용되고 있고, 개방된 교육 자료 활용을 목적으로 개발되어 보급된 사례는 극히 드물다. 표 4~6에는 국가적 차원의 공공기관인 한국교육개발원에서 정책적으로 개발한 중등 정보과학 영재 교수학습 자료[9]를 보였는데, 이 분야의 정

표 4. 탐색단계 교수학습 자료 제목
Table 4. Coursework Titles for Groping Phase

컴퓨터와 정보 외의 관계 찾기	1주 정보사회와 정보산업 2주 부울대수와 논리회로 3주 중앙처리장치의 기능과 동작원리 4주 주기억장치의 기능과 동작원리 5주 컴퓨터의 활용분야 찾기 6주 저작권 침해방지 및 대응 방안 7주 사생활 침해 방지 및 대응 방안 8주 문자데이터 표현과 처리방법 9주 수치데이터 표현과 처리방법 10주 소리데이터 표현과 처리방법
미래 의컴퓨터 기술	1주 전자태그의 동작원리 이해하기 2주 개인정보 침해 사례 분석하기 3주 전자태그 활용 사례 탐구하기 4주 유비쿼터스 홈 시스템 체험하기 5주 헬스케어 서비스 탐구하기 6주 새로운 유비쿼터스 서비스 제안하기 7주 휴대목적에 따른 기능 분석하기 8주 휴대방법에 따른 디자인 제안하기 9주 새로운 개인용 휴대장치 설계하기 10주 미래의 일상생활 상상하기
미래 의가전제품	1주 스스로 이동하는 기능 탐구하기 2주 사람과 대화하는 기능 탐구하기 3주 상황을 판단하는 기능 탐구하기 4주 미래의 가정용 로봇 설계하기

표 5. 발전단계 교수학습 자료 제목
Table 5. Coursework Titles for Developing Phase

알고리즘 이해하기	1주 알고리즘 이야기 2주 순서도를 이용한 알고리즘 표현하기 3주 의사코드를 이용한 알고리즘 표현하기 4주 생활 속 알고리즘 구현하기(등교 알고리즘) 5주 생활 속 알고리즘 구현하기(수업 알고리즘) 6주 속 알고리즘 구현하기(점심시간 알고리즘) 7주 생활 속 알고리즘 구현하기(하교 알고리즘) 8주 생활 속 알고리즘 구현하기(소풍 알고리즘) 9주 생활 속 알고리즘 구현하기(긴급 상황처리 알고리즈) 10주 보고서 작성
놀이공원 알고리즘 구현하기	1주 자료구조 이야기 2주 배열과 행렬을 이용한 자료구조 표현 3주 트리의 표현 및 순회방법 4주 집에서 공원까지 그래프 형태로 표현하기 5주 집에서 놀이공원까지 최단거리 구현하기 6주 공원의 구성을 그래프로 구현하기 7주 놀이기구 정렬 방법 구현하기 8주 놀이기구 탐색 방법 구현하기 9주 새로운 놀이기구를 구현하여 설치하기 10주 놀이공원의 알고리즘 구현하기(보고서 작성)
논리회로 이해하기	1주 논리 회로와 표현 방법 2주 논리 대수학 이해하기 3주 논리 회로 함수 표현하기 4주 논리 회로 설계하기

표 6. 심화단계 교수학습 자료 제목
Table 6. Coursework Titles for Deepening Phase

정보통신 이해하기	1주 정보통신 및 네트워크 이야기 I 2주 정보통신 및 네트워크 이야기 II 3주 인터넷 서비스 이야기 4주 새로운 정보통신 서비스 이야기 5주 차세대 인터넷 서비스 이야기 6주 통신 알고리즘 분석하기 I 7주 통신 알고리즘 분석하기 II 8주 통신 알고리즘 설계하기 9주 다중 접속 프로토콜 분석하기 10주 이더넷의 접속 프로토콜 구현하기
새로운 네비게이션 만들기	1주 텔레매틱스 이야기 2주 네비게이터의 특징 및 기능 이해하기 3주 네비게이터의 길찾기 서비스 원리 분석하기 4주 그래프 이론(한붓그리기, 오일러길) 5주 그래프 이론(해밀턴길) 및 합리적인 길찾기 6주 길찾기 알고리즘을 적용한 여행계획 설계 7주 텔레매틱스를 이용한 네비게이터의 인터페이스 특성 알아보기 8주 새로운 네비게이터 기능 제안하기 9주 새로운 네비게이터 기능 구현하기 10주 새로운 네비게이터 기능 개선 보고서 작성
교착상태 이해하기	1주 오리엔테이션 및 운영체제 2주 프로세스와 자원 3주 교착상태 이야기 I 4주 교착상태 이야기 II

책적 교재로는 거의 유일한 것이다. 이 교재는 중등 정보과학 영재교육 단계를 탐색, 발전, 심화로 분류하고, 각 단계의 단원 주제별 교육 자료를 제시하고 있다.

위 교수학습 자료에서 보는 바와 같이, 중등 교과 내용과 연계되어 개발된 교육 자료가 전무하고, 대부분이 정보 관련 기초 소양이나 정보시스템 요소 기술, 그리고 컴퓨터과학 지식(이론)을 이해하는 데에 치우쳐 있다. 이와는 다르게, 이 연구에서는 중등 수학 교과의 1차 연립 방정식 풀이를 컴퓨터 프로그래밍으로 해결하는 방안들을 단계적으로 확장하면서 진행하는 과정을 학습 자료로 개발하여 정보처리를 위한 핵심 요소인 창의적 문제발견과 문제해결 능력 배양을 위한 교수학습 자료의 보급에 기여한다.

III. 컴퓨터과학에서 행렬의 필요성과 활용

1. 2원 1차 연립방정식의 정의 및 풀이 방법

1.1 방정식의 정의

방정식(方程式)이란 식에 포함된 한 개 이상의 미지수(변수)들이 특정 값을 가질 때에만 참이 되는 식으로서, 그 미지수들이 어떤 값을 가져도 참이 되는 항등식(恒等式)과 대비된

다[10]. 이러한 방정식 중, 2 개 이상의 미지수들이 포함되고, 여러 개의 식으로 이루어진 방정식을 연립방정식이라 하는데, 특히 모든 미지수의 차수가 1일 경우에는 1차 연립방정식이라 부른다. 또한 미지수의 개수가 n일 때 n원 1차 연립방정식이라 부르는데, 현재 중등 수학에서는 2원 1차 연립방정식을 다룬다. 그림 2에 2원 1차 연립방정식의 예를 보였는데 이들의 해는 모두 동일하게 x는 2, y는 3이다.

```
예 1)  2x -  y =  1  ... ①
       3x +  y =  9  ... ②
예 2)   x +  y =  5  ... ③
       3x - 2y =  0  ... ④
예 3)  2x + 2y = 10  ... ⑤
       3x + 6y = 24  ... ⑥
```

그림 2. 2원 1차 연립 방정식의 예
Fig. 2. Examples of Equations

1.2 연립 방정식의 풀이 방법

1차 연립방정식의 풀이 과정은 여러 개의 식으로부터 미지수를 하나씩 소거해나감으로써 궁극적으로는 미지수가 하나인 방정식을 얻은 후 해를 얻는 과정을 반복한다. n(n>2)원 1차 연립방정식 풀이의 기본이 되는 2원 1차 연립방정식의 풀이 과정은 크게 가감법, 대입법, 등치법으로 요약될 수 있는데, 연립방정식의 모양에 따라 가장 적절한 방법을 선택하면 되는 것이지, 꼭 특정 방법을 적용해야 되는 것은 아니다.

■ 가감법

가감법은 두 방정식의 양변을 더하거나 감함으로써 단번에 미지수 하나를 소거하는 방법으로 가장 간단한 경우에 해당된다. 그림 2의 연립방정식 예 1)에서 두 식 ①과 ②의 양변을 더하면 미지수 y가 소거된 아래의 방정식이 얻어지고 이로부터 x 값 2가 구해진다.

$5x = 10.$

■ 대입법

대입법은 두 식 중 하나에서 미지수 하나(이를테면 x)를 다른 미지수(이를테면 y)와 상수로 표현(정렬)한 다음, 그 결과를 다른 식의 미지수(x) 부분에 대입함으로써 미지수 하나를 소거하는 효과를 얻는 방법이다. 그림 2의 연립방정식 예 2)는 대입법 풀이에 적당하다. 우선, 식 ③을 미지수 x를 기준으로 재정렬하면 다음 식을 얻을 수 있다.

$x = -y + 5.$

위 식을 식 ④의 x 자리에 대입하면 미지수 x가 소거되고 y만 남아있는 아래의 식을 얻을 수 있고, 이 식으로부터 y 값 3이 얻어진다.

$3(-y + 5) - 2y = 0.$

■ 등치법

등치법은 두 식 모두를 동일한 미지수(이를테면 x)를 다른 미지수(이를테면 y)와 상수로 표현한 다음, 두 식을 같게 놓아 미지수 하나(x)를 소거한 다음, 다른 미지수(y)를 구한다. 그림 2의 방정식 예 3)에서 먼저, 두 식 ⑤와 ⑥을 각각 미지수 x를 미지수 y와 상수로 재정렬하면 아래의 두 식이 얻어진다.

$x = -y + 5$ ... ⑤'

$x = -2y + 8$ ... ⑥'

위 두 식 ⑤'와 ⑥'의 우변은 동일하므로 이 두식을 아래와 같이 같게 놓은 방정식으로부터 y 값 3을 구할 수 있다.

$-y + 5 = -2y + 8.$

2. 컴퓨터 기초 활용에 의한 연립방정식 풀이

컴퓨터를 이용하여 어떤 문제를 해결하기 위해서는, 그 문제를 컴퓨터 프로그래밍 언어로 재구성하여야 한다. 프로그래밍 언어는 변수와 실행 구문으로 구성되어 있기 때문에 해결하고자 하는 문제를 프로그래밍 언어로 표현하는 일 자체가 하나의 창작 활동이다. 여기서는 C 언어를 기반으로 가감법, 대입법, 그리고 대치법 각각의 풀이 과정을 유도하도록 한다.

2.1 연립방정식의 표현

연립방정식의 두 식은 그림 2와 같이 ax+by=c 형태로 서술하기로 한다면, 프로그래밍 언어에서는 미지수 x와 y의 계수 a, b와 상수 c만을 적절히 다루면 된다. 따라서, 두 식의 계수와 상수를 저장할 변수로 다음과 같이 6 개의 변수를 도입하면 2원 1차 연립방정식의 표현이 가능하다(단, 첫 번째 식을 식 ❶, 두 번째 식을 식 ❷라 칭함).

-cx1:식 ❶에서 x의 계수, -cy1:식 ❶에서 y의 계수
-co1:식 ❶,에서의 상수, -cx2:식 ❷에서 x의 계수
-cy2:식 ❷에서 y의 계수, -co2:식 ❷에서의 상수

이를테면 그림 2의 연립방정식 예 1)은 아래의 그림 3과 같은 C 언어 코드로 표현될 수 있다.

```
void main()
{       /*    □x  +    □y    =    □ */
    float  cx1 = 2, cy1 = -1, co1 = 1;…❶
    float  cx2 = 3, cy2 =  1, co2 = 9;…❷
    :
}
```

그림 3. 연립방정식의 C 언어 표현
Fig. 3. Representation of Equations in C Language

2.2 가감법 풀이 과정

가감법을 적용하기 위해서는 두 미지수 중 어느 한 미지수의 계수 절대 값이 두 식 사이에서 일치해야 한다. 그림 3에서 보는 바와 같이, 육안으로는 어느 쪽 계수가 일치하는지를 쉽게 알 수 있으나, 컴퓨터에서는 임의의 연립방정식에 대하여 어느 쪽 계수가 일치할지를 단정 지을 수 없다. 따라서, 한 가지 방법으로 두 식의 양변에 대한 합을 구하여 0이 되는 계수가 나오면 그 상태에서 해를 구하고, 0이 되는 계수가 존재하지 않으면 두 의 양변에 대한 차를 구한 후 해를 구하는 방안이 가능하다. 물론 계수가 0인 미지수는 두 미지수 중 어느 하나일 것이다. 이 방안에 대한 해결 절차의 개념도를 그림 4에 보였다.

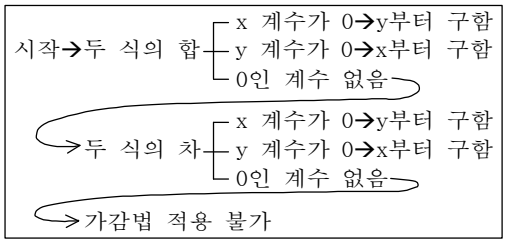

그림 4. 가감법 절차의 개념도
Fig. 4. Conceptual Procedure for Addition and Subtraction Method

[단계 1] 위 그림 4의 절차에 프로그래밍 언어를 적용하기 위해 먼저 필요한 사항은 무엇일까? 다름 아닌 두 식의 합이나 차를 표현하는 방법인데, 이 식들 또한 □x+□y=□ 형태이므로 이 식의 계수와 상수를 각각 변수 cx3, cy3, co3에 대응시키도록 한다(그림 5의 ❸).

```
void main()
{   /*     □x + □y  = □ */
   float  cx3, cy3, co3;   ……❸
   :
   cx3=cx1+cx2; cy3=cy1+cy2; co3=co1+co2;
   :
   cx3=cx1-cx2; cy3=cy1-cy2; co3=co1-co2;
   :
}
```

그림 5. 두 식의 가·감을 위한 C 언어 코드
Fig. 5. C code for Addition and Subtraction of Two Equations

[단계 2] 도입된 변수를 바탕으로 두 식의 합과 차를 구한다. 이 절차를 위한 C 언어 코드 예를 그림 5에 보였다.

[단계 3] 가·감에 의해 도출된 소거식에 계수 0인 미지수가 존재하는지와 존재한다면 어느 미지수의 계수가0인지에 대한 경우의 수를 식별하는 흐름제어 구조를 그림 4에 의거 모색한다. 그림 6에 이를 위한 C 언어 코드 예를 보였다.

```
void main()
{
   float   x, y;   /* 해 */
   :
   if (cx3 == 0 || cy3 == 0) {
      if (cx3 != 0)
         x = co3/cx3, y=(co1-cx1*x)/cy1;
      else
         y = co3/cy3, x=(co1-cy1*y)/cx1;
  }
   :
}
```

그림 6. 경우의 수 식별을 위한 C 언어 코드
Fig. 6. C code for Handling Cases

[단계 4] 단계 1-3을 연계하여 문제해결 절차를 완성한다. 그림 7에 가감법 풀이과정을 위한 전체 C 언어 코드 예를 보였다.

```
void main()
{   /*   □x   +   □y    =   □ */
  float  cx1 = 2, cy1 = -1, co1 = 1;…❶
  float  cx2 = 3, cy2 =  1, co2 = 9;…❷
  float  cx3,     cy3,      co3;……❸
  float   x, y;   /* 해 */
  cx3=cx1+cx2; cy3=cy1+cy2; co3=co1+co2;
  if (cx3 != 0 && cy3 != 0)/*덧셈후뺄셈시도*/
    cx3=cx1-cx2, cy3=cy1-cy2, co3=co1-co2;
  if (cx3 == 0 || cy3 == 0) {
    if (cx3 != 0)
       x = co3/cx3, y=(co1-cx1*x)/cy1;
    else
       y = co3/cy3, x=(co1-cy1*y)/cx1;
    printf("x=%7.2f, y=%7.2fWn", x, y);
  }
  else
    printf("가감법 적용 불가Wn");
}
```

가능 / 불가

그림 7. 가감법을 위한 C 언어 코드
Fig. 7. C code for Addition and Subtraction Method

2.3 대입법 풀이 과정

[단계 1] 대입법에서는 일차적으로 두 방정식 중 하나로부터 x=□y+ 혹은 y=□x+□ 형태의 변환 식을 얻어야 하는데, 여기서는 첫 번째 식으로부터 x=□y+□ 형태의 변형식을 얻는 것으로 하고, 이 식의 표현을 위해미지수 y의 계수와 상수에 각각 cy4, co4 명칭의 변수 를 대응시킨다(그림 8의 ❹). 그림 3의 주어진 연립방정식으로부터 cy4와 co4를 구하는 C 언어 코드는 그림 8과 같다.

```
void main()
{    /* x = □y + □ */
     float  cy4, co4; ·····❹
     :
     cy4 = - cy1 / cx1;
     co4  = co1 / cx1;
     :
}
```

그림 8. 첫 번째 식의 변형을 위한 C 언어 코드
Fig. 8. C code for Transformation of the First Equation

[단계 2] 1 단계 변형식 ❹를 연립방정식의 두 번째 식 ❷에 대입하면 식 ❷는 △y+△=△ 형태의 1차 방정식이 얻어지는데, 이 식의 오른 쪽 항의 상수는 식 ❷의 오른쪽 항 co2 그대로다. 이 식을 cy5, co5, co5′ 등 세 개의 변수를 도입하여 표현하기로 하면(그림 9의 ❺), 이 식을 그림 8의 식 ❹로부터 얻기 위한 C 언어 코드 예는 그림 9와 같다.

```
void main()
{       /* □y  + □  = □ */
     float  cy5, co5, co51; ·····❺
     :
     cy5 = cy4*cx2 + cy2;
     co5 = co4*cx2;  co51 = co2;
     :
}
```

그림 9. 대입에 의한 미지수 소거를 위한 C 언어 코드
Fig. 9. C code for Reduction of Equation by Substitution

[단계 3] 그림 9의 일차 방정식 ❺로부터 y 값이 결정되고, 그 y 값을 두 식에 중 하나에 적용하여 x 값을 얻을 수 있다. 여기서는 식 ❶을 사용하기로 한다. 그림 10은 y와 x 값을 얻는 C 언어 코드이고, 그림 11은 단계 1~3을 통합한 완성된 코드이다.

```
void main()
{
     float   x, y;
     :
     y = (co51 - co5) / cy5;
     x = (co1 - cy1 * y) / cx1;
     :
}
```

그림 10. 소거된 식에서 해를 구하는 C 언어 코드
Fig. 10. C code to Resolve the Reduced Equation

2.4 등치법 풀이 과정

등치법에서 해야 할 첫 번째 일은, 방정식의 두 식에서 동일한 미지수 하나을 선택하여 그 미지수를 다른 미지수와 상수로 표현하는 일이다.

[단계 1] 미지수 x를 소거하기 위해 두 식을 각각 x=□y+□, x=△y+△ 형태로 변형시킨다. 첫 번째 식의 표현을 위해 cy6, co6(그림 12의 ❻), 두 번째 식의 표현을 위해 cy7, co7(그림 12의 ❼)의 변수를 도입하면, 이들은 그림 12의 C 언어 코드로 얻어질 수 있다.

[단계 2] 두 식의 오른쪽 항을 같게 놓은 □y+□=△y + △ 형태의 방정식으로부터 y 값을 얻는다. 이 과정을 위한 C 언어 코드 예를 그림 13에 보였고, 그림 14에 등치법을 위한 C 언어 코드 전체를 보였다.

```
#include <stdio.h>
void main() /*   2x     -  1y     =  1 */
{           /*   3x     +  1y     =  9 */
   float  cx1 = 2,  cy1 = -1, co1 = 1;
   float  cx2 = 3,  cy2 =  1, co2 = 9;
   float  cy4, co4;       /* x = □y + □ */
   float  cy5, co5, co51; /* Δy + Δ = Δ  */
   float  x, y;
   cy4 = -cy1 / cx1;        /* 단계 1 */
   co4 = co1 / cx1;
   cy5 = cy4 * cx2 + cy2;   /* 단계 2 */
   co5 = co4 * cx2;
   co51 = co2;
   y = (co51 - co5) / cy5;  /* 단계 3 */
   x = (co1 - cy1 * y) /cx1;
   printf("x=%7.2f, y=%7.2f₩n", x, y);
}
```

그림 11. 대입법을 위한 C 언어 코드
Fig. 11. C code for Substitution Method

```
void main()
{
    float  cy5, co5; /* x = □y + □ */…❻
    float  cy6, co6; /* x = Δy + Δ */…❼
    :
    cy6 = - cy1 / cx1;
    co6 = co1 / cx1;
    cy7 = - cy2 / cx2;
    co7 = co2 / cx2;
    :
}
```

그림 12. 공통 미지수 기준 재정렬 C 언어 코드
Fig. 12. C code for Rearrangement of Equations based on a Common Unknown

```
void main()
{
     float  y;
     :
     y = (co7 - co6) / (cy6 - cy7);
     :
}
```

그림 13. 등치식에서 y 값을 구하는 C 언어 코드
Fig. 13. C code to Resolve the Rearranged Equation

```
#include <stdio.h>
void main()  /*   2x     -  1y     =  1 */
{            /*   3x     +  1y     =  9 */
     float  cx1 = 2, cy1 = -1,  co1 = 1;
     float  cx2 = 3, cy2 = +1,  co2 = 9;

     float  cy6, co6;  /*  □y + □  */
     float  cy7, co7;  /*   Δy + Δ  */
     float  x, y;

     cy6 = - cy1 / cx1;  /* 단계 1 */
     co6 = co1 / cx1;
     cy7 = - cy2 / cx2;
     co7 = co2 / cx2;
                         /* 단계 2 */
     y = (co7 - co6) / (cy6 - cy7);
     x = (co1 - cy1 * y) /cx1;

     printf("x=%7.2f, y=%7.2fWn", x, y);
}
```

그림 14. 등치법을 위한 C 언어 코드
Fig. 14. C code for Equalization Method

3. 행렬을 이용한 연립방정식의 표현 및 풀이

3.1 행렬의 개념 및 기본 규칙

행렬이란 그림 15와 같이 문자나 숫자가 사각 형태 즉, 행(좌우로 동일 선상에 나열된 것 들)과 열(상하로 동일 선상에 나열된 것 들)로 구성된 것을 말하는데, 그 활용은 사용자의 정의에 따라 다양하다. 행렬 중 줄의 개수와 열의 개수가 같은 것을 n차 정방 행렬(n은 줄과 열의 개수)이라 부른다. 수학에서 다루는 행렬 또한 나름대로의 정의와 규칙을 가지고 있는데, 여기서는 몇 가지 사례를 기반으로 일반 수학에서 다루는 행렬에 대한 이해를 발전시켜나가기로 한다. 보통 행렬은 단독으로 사용되는 경우보다는 2 개 이상의 행렬이 관계를 맺음으로써 의미를 갖는다.

$$\begin{pmatrix} 10 & 20 & 30 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 3 \\ a & b & c \end{pmatrix}$$

$$\begin{pmatrix} 3 \\ 7 \\ 6 \end{pmatrix} \quad \begin{pmatrix} a & x \\ b & y \end{pmatrix}$$

그림 15. 행렬의 예
Fig. 15. Examples of Matrix

■ 행렬 곱셈 규칙

점보 문구점에서 노트 1 권 가격은 150원, 연필 1 자루는 100원에 판매하고, 영철이가 점보 문구점에서 노트 2 권과 연필 3 자루를 샀다면, 그 장부를 그림 16의 예 1)과 같이 행렬 관계를 이용하여 표현할 수 있다. 예 2)는 같은 문구점에서 철수가 노트 4권, 연필 1자루를 구입했을 때 철수 자료가 추가된 모습이다.

점보 문구점 외에 럭키 문구점이 있어서, 그 곳에서는 노트와 연필을 각각 130원과 120원에 판매하고, 영철이가 두 문구점에서 각각 노트 2 권과 연필 4 자루를 구입했다면 그 장부는 예 3)과 같이 표현된다. 예 4)는 철수도 영철이와 동일하게 노트와 연필을 두 문구점에서 구입했을 때의 표현을 보였다. 이들 예에서 보인 행렬 관계가 성립하기 위해서는 문구점에서 파는 품목의 개수(왼쪽 행렬에서 열의 개수)와 구입한 사람의 품목의 개수(오른쪽 행렬의 줄의 개수)가 동일 즉, 1:1 대응을 이뤄야 되고, 그 결과 행렬은 품목의 개수와는 관계없이 문구점의 수(행의 수)와 구매자의 수(열의 수)

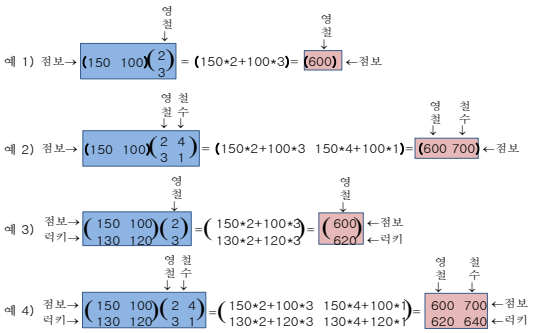

그림 16. 행렬의 곱셈 개념
Fig. 16. Concept of Matrix Product

만큼의 행과 열로 이루어진다는 사실을 알 수 있다. 이와 같은 행렬의 관계를 행렬의 곱(product)이라 부른다. 그림 16의 행렬 곱셈 규칙에 의하면, 곱셈이 가능한 임의의 세 행렬 A, B, C를 곱할 때, (AB)C의 결과와 A(BC)의 결과가 일치하는 결합 법칙이 성립한다는 사실도 알 수 있다.

■ 행렬 가감 규칙

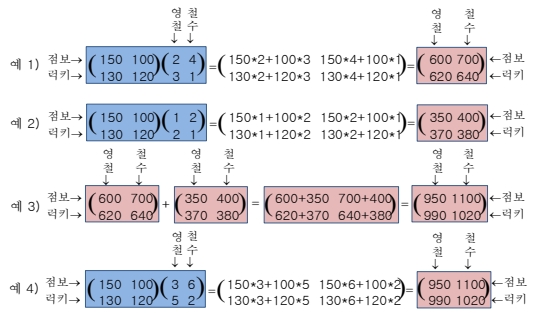

그림 17. 행렬의 덧셈 개념
Fig. 17. Concept of Matrix Addition

영철이와 철수가 그림 17의 예 1)과 같이 점보 문구점과 럭키 문구점에서 각각 노트 2 권, 연필 3 자루와 노트 4 권,

연필 1 자루를 구입한 후, 예 2)와 같이 영철이는 노트 1 권, 연필 2를 철수는 노트 2 권, 연필 1 자루씩을 더 구입한 최종 결과는 예 3)의 표현으로 구한다. 이 결과는 예 4)의 전체 구입 개수에 의한 행렬의 곱셈으로 얻은 결과와 일치한다. 이와 같은 행렬 관계를 행렬의 합이라 하는데, 행렬의 합이 이루어지기 위해서는 두 행렬의 모양이 정확이 일치해야 함을 알 수 있다. 행렬의 합과 동일하게 행렬의 뺄셈 규칙도 성립한다.

■ 행렬의 실수배

그림 18의 예 1)은 영철이와 철수가 점보 문구점과 럭키 문구점에서 노트와 연필을 2 권, 3자루와 4 권 1 자루를 각각 구입한 수량의 표현이다. 예 2)는 영철이와 철수가 똑 같은 수량으로 3 번 구입했을 때의 수량 표현인데, 이를 행렬의 실수배 형태로 표현하면 예 3)과 같다.

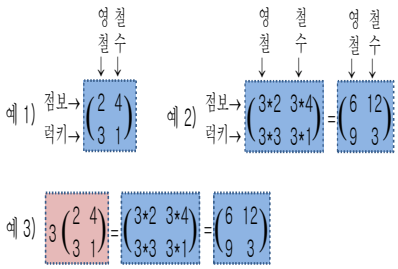

그림 18. 행렬의 실수배 개념
Fig. 18. Concept of Multiplied Matrix with Real Number

3.2 행렬을 이용한 연립방정식의 표현 및 풀이 과정

■ 행렬식을 이용한 연립 방정식 표현

그림 16에 보인 행렬 곱셈 규칙을 응용하면 그림 2의 연립방정식들을 행렬식으로 표현할 수 있다. 그림 19에 행렬식으로 표현되는 연립방정식의 예를 보였는데, 미지수 x, y의 계수만을 따로 떼어 계수행렬로 모으고, 미지수 x, y 만을 모아 미지수 행렬을 만들며, 상수항만을 모아 상수행렬을 만든 후, 좌변에 계수행렬과 미지수 행렬의 곱을, 우변에는 상수행렬을 두면 이 행렬식이 바로 연립방정식과 동치가 됨을 알 수 있다.

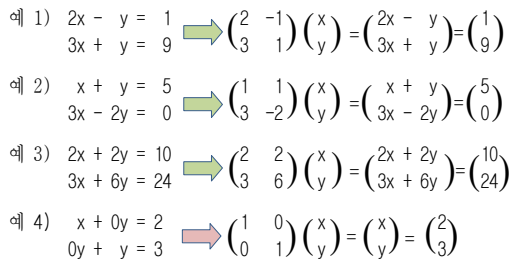

그림 19. 행렬을 이용한 연립방정식 표현
Fig. 19. Expression of Equations by Matrix

■ 행렬식으로 표현된 연립방정식의 풀이 방법 착안

그림 19의 예 4)를 자세히 살펴보면 이는 연립방정식이라기보다는 방정식의 해를 행렬로 표현한 것임을 깨달을 수 있다. 이 행렬식에서의 계수행렬은 좌상·우하 대각선에 위치한 원소는 모두 1이고, 나머지 원소는 모두 0인데, 이러한 행렬을 단위행렬이라 부른다. 어떤 행렬 A와 단위행렬 I의 곱인 AI 혹은 IA의 결과는 행렬의 곱셈 규칙에 따라 그대로 A이다. 따라서, 그림 19의 예 1) ~ 예 3)의 행렬을 적절하게 변형하여 예 4)와 같이 계수행렬이 단위행렬을 이루도록 재정렬할 수 있다면 결국 x, y 값이 얻어진다. 즉, 행렬로 표현된 연립방정식 양변에 적당한 연산을 가하여, 좌변의 계수행렬이

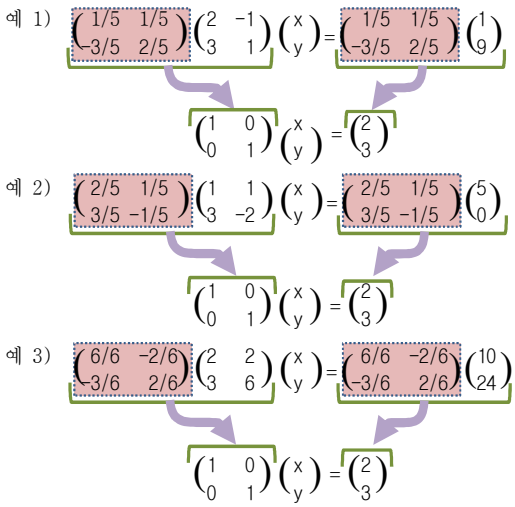

그림 20. 역행렬을 이용한 연립방정식 풀이 개념
Fig. 20. How to Resolve Equation using Reverse Matrix

단위행렬이 되도록 유도하면 우변에는 연립방정식의 해를 의미하는 행렬이 남게 되는 것이다.

■ 역행렬을 이용한 연립방정식의 풀이

그림 20에, 그림 19의 예 1) ~ 예 3)의 연립방정식 각각의 계수행렬들을 곱셈 규칙에 의해 단위행렬로 변화시킬 수 있는 대응행렬을 고안하여 해를 구하는 과정을 보였다. 그림 20에서 보는 바와 같이 주어진 연립방정식의 해를 구하는 한 가지 방법은 곧 계수행렬에 곱하여 단위행렬을 낳게 하는 행렬을 찾는 문제로 귀결되는데, 이 같은 행렬을 역행렬이라 부른다. 즉, 임의의 주어진 2차 정방행렬 $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$에 대한 역행렬을 $\begin{pmatrix} x & y \\ z & u \end{pmatrix}$라 하면 아래의 행렬식이 만족한다.

$$\begin{pmatrix} x & y \\ z & u \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

위 식을 항상 만족하게 하는 a, b, c, d 값을 x, y, z, u로 나타내면 그 것이 곧 역행렬이다. a, b, c, d 값을 찾는 과정은 아래와 같다.

[단계 1] 우선, 행렬의 곱셈 규칙에 따라 아래의 4 식을 얻는다.

xa+yc=1……① xb+yd=0……②

za+uc=0 ……③ zb+ud=1 ……④

[단계 2] 식 ①의 양변에 b를 곱한 식 xab+ybc=b에서, 식 ②의 양변에 a를 곱한 식 xab-yad=0을 빼면 즉, ①*b-②*a를 양변 계산하면 다음식이 얻어진다.

ybc-yad=b ⇒ y(bc-ad)=b ⇒ y =-b/(ad-bc) …⑤

[단계 3] 마찬가지 방법으로 ①*d-②*c를 양변끼리 계산하면 다음식이 얻어진다.

xad-xbc=d ⇒ x(ad-bc)=d ⇒ x=d/(ad-bc) ……⑥

[단계 4] ③*b-④*a를 양변끼리 계산하면 다음식이 얻어진다.

ubc-uad=-a ⇒ u(bc-ad)=-a ⇒ u=a/(ad-bc)….⑦

[단계 5] ③*d-④*c를 양변끼리 계산하면 다음식을 얻을 수 있다.

zad-zbc=-c ⇒ z(ad-bc)=-c ⇒ z=-c/(ad-bc)…⑧

위 결과를 요약하면, 임의의 2차 정방행렬 $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$에 대한 역행렬은 아래와 같이 정의된다(아래 식에서 행렬 위 -1 표시는 역행렬임을 의미함). 이 때, 유의해야 할

$$11x + 4y = 4,\quad 3x + 1y = 1 \Rightarrow \begin{pmatrix} 11 & 4 \\ 3 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$$

$$1/(11*1-4*3)\begin{pmatrix} 1 & -4 \\ -3 & 11 \end{pmatrix}\begin{pmatrix} 11 & 4 \\ 3 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = 1/(11*1-4*3)\begin{pmatrix} 1 & -4 \\ -3 & 11 \end{pmatrix}\begin{pmatrix} 4 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

그림 21. 역행렬을 이용한 연립방정식 풀이 예
Fig. 21. An Example of Application of Reverse Matrix to Resolve Equations

사항은 (ad-bc) 값이 0이면 역행렬(해)이 존재하지 않는다는 점이다.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \begin{pmatrix} d/(ad-bc) & -b/(ad-bc) \\ -c/(ad-bc) & a/(ad-bc) \end{pmatrix} = 1/(ad-bc)\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

그림 21에 역행렬을 이용한 연립방정식 풀이 과정의 예를 보였다.

3.3 컴퓨터에 의한 역행렬을 이용한 연립방정식 풀이

[단계 1] 미지수 x, y위 계수 및 상수항을 행렬 구조와 동일한 2차원 배열 변수로 표현한다. 그림 22는 그림 21의 연립방정식을 배열로 표현한 C 언어 코드 예다.

```
void main()          /* 11x + 4y = 4 */
{                    /*  3x + 1y = 1 */
   float  co[2][3] = { 11,   4,   4,
                        3,   1,   1 };
     :
}
```

그림 22. 연립방정식의 행렬 표현을 위한 C 언어 코드
Fig. 22. C code for Representation of Equations in Matrix

[단계 2] 역행렬이 존재하는지를 판별하기 위해 D=(ad-bc)의 값이 0인지를 조사하여 만약 0이면 해가 없기 때문에 프로그램 진행을 중단한다. 그림 23은 역행렬을 구하는 C 언어 코드의 예이다.

```
void main()
{
  :
  float  rc[2][2];     /* 역행렬 저장용 */
  float  D;
  D = co[0][0]*co[1][1] - co[0][1]*co[1][0];
  if (D == 0) {
     printf("해가 없음...₩n);
     exit(1);
  }
  rc[0][0]= co[1][1]/D; rc[0][1]=-co[0][1]/D;
  rc[1][0]=-co[1][0]/D; rc[1][1]= co[0][0]/D;
  :
}
```

그림 23. 역행렬 적용하는 C 언어 코드
Fig. 23. C code for Application of Reverse Matrix

[단계 3] 역행렬을 상수항 행렬(co[0][2], co[1][2])에 곱하여 최종 해를 구한다. 이 부분을 위한 C 언어 코드 예를 그림 24에 보였고, 그림 25에는 위 전체 과정을 통합해 보였다.

```
void main()
{
   :
   float  x, y;
   x = rc[0][0]*co[0][2] + rc[0][1]*co[1][2];
   y = rc[1][0]*co[0][2] + rc[1][1]*co[1][2];
   printf("x=% 2.2f, y=% 2.2f₩n", x, y);
}
```

그림 24. 역행렬로 해를 구하는 C 언어 코드
Fig. 24. C code to Resolve Equations by Reverse Matrix

3.4 가우스 소거에 의한 연립방정식 풀이

역행렬을 이용한 연립방정식의 풀이의 한계는 무엇일까? 미지수의 개수가 늘어나면 간단한 공식에 의한 역

```
#include <stdio.h>
void main()          /* 11x + 4y = 4  */
{                    /*  3x + 1y = 1  */
  float  co[2][3] = {  11,   4,   4,
                        3,   1,   1 };
  float  rc[2][2];   /* 역행렬 저장용 */
  float  D;
  float  x, y;

  D = co[0][0]*co[1][1] - co[0][1]*co[1][0];
  if (D == 0) {
      printf("해가 없음...₩n");
      exit(1);
  }
  rc[0][0]= co[1][1]/D; rc[0][1]=-co[0][1]/D;
  rc[1][0]=-co[1][0]/D; rc[1][1]= co[0][0]/D;

  x = rc[0][0]*co[0][2] + rc[0][1]*co[1][2];
  y = rc[1][0]*co[0][2] + rc[1][1]*co[1][2];

  printf("x=%7.2f, y=%7.2f₩n", x, y);
}
```

그림 25. 역행렬에 의한 연립방정식 풀이 C 언어 코드
Fig. 25. C code to Resolve Equations by Reverse Matrix

행렬 구하기가 쉽지 않다. 이를 해결하기 위해 가우스 소거법을 사용한다.

■ 삼각행렬과 1차 연립방정식

삼각행렬이란 그림 26과 같이 정방행렬 중에서 대각선을 기준으로 아래쪽이나 위쪽 원소들이 모두 0인 행렬을 말하는데, 아래쪽이 0인 것을 상삼각행렬, 위쪽이 0인 것을 하삼각행렬이라 부른다(여기서는 상삼각행렬을 삼각행렬이라 하기로 함). 만약 1차 연립방정식의 계수행렬이 삼각행렬 형태로 구성되어 있다면 그 방정식의 해를 구하는 일이 어렵지 않다. 이를테면, 계수행렬이 삼각행렬을 이루고 있는 아래의 3원 1차 연립 방정식에서, 우선 식 ③으로부터 z 값 2를 얻고, 이 값을 식 ②에 대입하여 y 값 1을 얻을 수 있으며, 마지막으로 이들 z와 y 값을 이용해서 식 ①로부터 x 값 3을 얻을 수 있다.

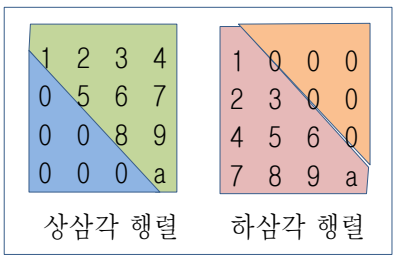

그림 26. 삼각행렬의 정의
Fig. 26. Definition of Triangular Matrix

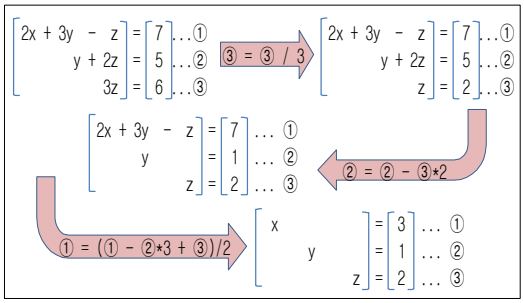

그림 27. 삼각행렬의 단위행렬 변환
Fig. 27. Transformation of Triangular Matrix to Element Matrix

$$2x + 3y - z = 7 \ \ldots\ldots\ ①$$
$$y + 2z = 5 \ \ldots\ldots\ ②$$
$$3z = 6 \ \ldots\ldots\ ③$$

```
#include <stdio.h> /* 2x + 3y -  z  = 7 */
void main()        /*       y + 2z  = 5 */
{                  /*           3z =  6 */
 float co[3][4] = {    2,   3,  -1,   7,
                       0,   1,   2,   5,
                       0,   0,   3,   6 };
 float r;
 int   i, j;

 co[2][3] = co[2][3]/co[2][2];          2  3 -1  7
 co[2][2] = co[2][2]/co[2][2];          0  1  2  5
                                        0  0 [1] [2]
 r = co[1][2];
 co[1][2] = co[1][2] - co[2][2]*r;      2  3 -1  7
 co[1][3] = co[1][3] - co[2][3]*r;      0  1 [0] [1]
 co[1][3] = co[1][3]/co[1][1];          0  0  ①  ②
 co[1][1] = co[1][1]/co[1][1];

 r = co[0][1];
 co[0][1] = co[0][1] - co[1][1]*r;      2 [0] -1  4
 co[0][3] = co[0][3] - co[1][3]*r;      0  ①  0  ①
                                        0  0  1  2
 r = co[0][2];
 co[0][2] = co[0][2] - co[2][2]*r;      2  0 [0] [6]
 co[0][3] = co[0][3] - co[2][3]*r;      0  1  0  1
                                        0  0  ①  ②
 co[0][3] = co[0][3]/co[0][1];
 co[0][1] = co[0][1]/co[0][1];

 for (i = 0; i < 3; i++) {
       for (j = 0; j < 4; j++)
           printf("%7.2f ", co[i][j]);
       printf("₩n");
 }
}
```

그림 28. 삼각행렬의 단위행렬 변환 C 언어 코드
Fig. 28. C code for Transformation of Triangular Matrix into Element Matrix

```
#include <stdio.h>

void main()
{
    float co[3][4] = { 2,  3, -1,  7,
                       0,  1,  2,  5,
                       0,  0,  3,  6 };
    float r;
    int   i, j, k;

    for (k = 2; k >= 0; k--) {
      for (i = k+1; i <= 2; i++) {
         r = co[k][i];
         co[k][i] = co[k][i] - co[i][i]*r;
         co[k][3] = co[k][3] - co[i][3]*r;
      }
      r = co[k][k];
      co[k][3] = co[k][3] / r;
      co[k][k] = co[k][k] / r;
    }

    for (i = 0; i <=2; i++) {
       for (j = 0; j <= 3; j++)
          printf("%7.2f ", co[i][j]);
       printf("Wn");
    }
}
```

그림 29. 삼각행렬의 단위행렬 변환 C 언어 코드(일반화)
Fig. 29. C code for Transformation of Triangular Matrix into Element Matrix(Generalized)

위 내용을 행렬 관점에서 관찰해보면, 식 ③에서부터 출발하여 차례차례 계수행렬의 대각 원소를 1로, 나머지 원소는 모두 0으로 만들어 나아가면 결국 계수행렬이 그림 19의 예 4)와 같은 단위행렬로 변환되므로 해

가 구해진다. 그림 27에는 삼각행렬인 위 3원 1차 연립방정식의 계수행렬을 단위행렬로 변환시키는 과정을, 그림 28에는 이를 위한 C 언어 코드 예를 보였다. 그림 29에는 미지수가 더 많은 경우에도 활용될 수 있도록 그림 28을 반복(루프) 문장을 사용하여 일반화시킨 C 언어 코드 예를 보였다.

■ 계수행렬의 삼각행렬 변환(가우스 소거)

계수행렬로부터 삼각행렬을 얻기 위해서는 두 번째 식부터 시작해서 앞 쪽의 미지수를 하나씩 소거해나감으로써 가능하다. 그림 30에 삼각행렬을 얻는 과정에 대한 예를 보였다. 이 예의 원래 연립방정식에서 식 ②의 미지수 x는, 식 ①의 모든 계수 및 우변 상수에 1/2을 곱한 후 두 식의 차를 구하면 소거할 수 있다. 이 때, 식 ①을 중심식, 식 ②를 소거식이라 부르기로 한다. 마찬가지 방법으로 소거식 ③의 미지수 x는 중심식 ①의 모든 계수 및 우변 상수에 3/2을 곱한 후 두 식의 차를 구하면 소거할 수 있다. 이 과정을 소거식 ④까지 적용하면 단계 1의 연립 방정식에서 보는 바와 같이 미지수 x가 소거된 식 ②, ③, ④를 얻을 수 있다. 이번에는 단계 1의 식 ②를 중심식으로 하여 소거식 ③과 ④에서 미지수 y를 소거하면 단계 2의 방정식 ③, ④가 얻어진다. 다음으로 단계 2의 방정식 ③, ④에서 중심식 ③을 토대로 소거식 ④의 미지수 y를 소거하면 단계 3의 방정식이 얻어지고, 결국 전체 연립방정식의 계수행렬은 삼각행렬을 이룬다. 이 과정에서 소거하고자 하는 미지수의 중심식 계수를 a, 소거식 계수를 b라 하면 중심식에 곱할 수는 b/a로 일반화되는데, 이를 적용한 C

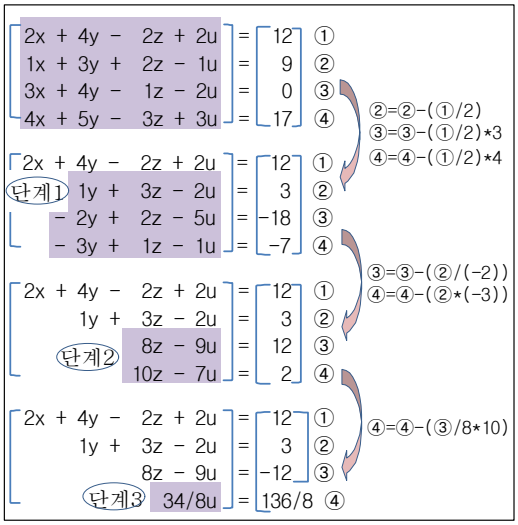

그림 30. 삼각행렬 변환 과정(가우스 소거)
Fig. 30. Process of Transformation into Triangular Matrix(Gaussian Elimination)

```
#include <stdio.h>
void main()
{
  float co[4][5] = { 2,   4,  -2,   2,  12,
                     1,   3,   2,  -1,   9,
                     3,   4,  -1,  -2,   0,
                     4,   5,  -3,   3,  17};
  float r;
  int   i, j, k;

  for (k = 0; k < 3; k++) {  /*단계,중심식*/
     for (i = k + 1; i < 4; i++) {/*소거식*/
        r = co[i][k] / co[k][k];
        for (j = k; j < 5; j++) /*계수,상수*/
           co[i][j] = co[i][j] - co[k][j]*r;
     }
  }

  for (i = 0; i < 4; i++) {
     for (j = 0; j < 5; j++)
        printf("%7.2f ", co[i][j]);
     printf("Wn");
  }
}
```

/*2x + 4y − 2z + 2u =12 */
/*1x + 3y + 2z − 1u = 9 */
/*3x + 4y − 1z − 2u = 0 */
/*4x + 5y − 3z + 3u =17 */

그림 31. 가우스 소거를 위한 C 언어 코드
Fig. 31. C code for Gaussian Elimination

언어 코드 예를 그림 31에 보였다. 그렇다면, 그림 31의 프로그램을 다수의 연립방정식 풀이에 시도했을 때, 그 해가 정확한지를 검증해보면 어떤 현상을 발견할 수 있을까?

3.5 유클리드 호제법

그림 31의 가우스 소거 프로그램이 진행할 때, 계수에 1/3, 3/11 등과 같은 무리수가 나타하면 정확한 해를 구할 수 없다. 왜냐하면 컴퓨터에서는 값을 저장하는 변수의 공간이 한정되어 있어서 소수 이하 자리수가 끝없이 이어지거나 그 길이가 한계를 벗어나면 반올림이나 절삭으로 수를 끊어야 하기 때문이다. 이를 해결하는 방안은 없을까? 소거하고자 하는 미지수의 중심식 계수와 소거식 계수를 두 계수의 최소공배수로 일치시켜 계산하면 소거 과정에서 계수의 무리수 출현을 예방할 수 있다. 이를 위해서는 두 정수의 최소공배수를 구할 수 있어야 한다.

■ 최소공배수(LCM) 구하는 방법

두 정수에 대한 최소공배수를 구하는 방법에는 여러 가지가 있을 수 있으나, 여기서는 최대공약수를 이용하는 방법을 이용하기로 한다. 왜냐하면 최대공약수를 컴퓨터 프로그램으로 구현할 수 있는 아주 훌륭한 방법이 알려져 있기 때문이다. 우선 두 정수 a, b의 최대공약수가 g라면 a, b는 다음의 식 ⓐ와 ⓑ로 각각 표시될 수 있다. 물론 이들 두 식에서 n1 ~ nn , m1 ~ mm의 모든 수들은 서로 소, 즉 서로 간에 약수와 배수 관계에 있지 않다.

$$a = n1e1 \times n2e2 \times \cdots \times nnen \times g \quad \cdots\cdots \text{ⓐ}$$

$$b = m1d1 \times m2d2 \times \cdots \times mmdm \times g \quad \cdots\cdots \text{ⓑ}$$

위 두 식 ⓐ와 ⓑ를 양변 곱하면 아래의 식 ⓒ를 얻을 수 있다.

$$a \times b = n1e1 \times n2e2 \times \cdots \times nnen \times m1d1 \times m2d2 \times \cdots \times mmdm \times g \times g \quad \cdots\cdots \text{ⓒ}$$

위의 식 ⓒ의 양변을 g로 나누면 아래의 식 ⓓ가 얻어지는데, 이 식의 우변을 살펴보면 그 것이 곧 두 정수 a, b의 최소공배수임을 알 수 있다.

$$a \times b / g = n1 \times n2 \times \cdots \times nn \times m1 \times m2 \times \cdots \times mm \times g \quad \cdots\cdots \text{ⓓ}$$

즉, 두 정수 a, b의 최소공배수는 두 수의 최대공약수가 주어지면 식 ⓓ에 의해 곧바로 구해진다.

■ 최대공약수 구하는 방법(유클리드 호제법)

유클리드 호제법은 기원전 300년경 과학자 유클리드에 의해 발명된 최대공약수 계산 방법으로 컴퓨터 프로그램으로 구현하기에 꼭 알맞다[11]. 유클리드 호제법의 최대공약수 구하는 절차 및 C 언어 코드를 그림 32에 보였다.

⊙ 큰 수를 작은 수로 나눈 나머지를 얻고(큰 수 1071를 작은 수 1029로 나눈 나머지는 42), 나머지가 0이면 작은 수가 곧 최대 공약수다.
⊙ 나머지가 0이 아니면, 작은 수를 큰 수로, 나머지를 작은 수로 옮겨놓고 위 과정을 반복한다(작은 수였던 1029를 큰 수로, 나머지였던 42를 작은 수로 놓음).
⊙ 큰 수 1029를 작은 수 42로 나눈 나머지 21이 0이 아니므로, 42를 큰 수로, 21을 작은 수로 놓는다.
⊙ 큰 수 42가 작은 수 21로 나눠지기 때문에, 21이 최대 공약수 이다.

```
int gcd(int  a,  int b)
{
 int  big, sml, r;

 if (a > b) big = a, sml = b;
 else       big = b, sml = a;

 while(1) {
    r = big % sml;

    if (r == 0) return(sml);

    big = sml, sml = r;
 }
}
```

그림 32. 유클리드 호제법 및 C 언어 코드
Fig. 32. Euclidean algorithm and C code

■ 유클리드 호제법을 적용한 가우스 소거

소거하고자 하는 미지수 계수의 최소공배수를 적용하는 가우스 소거 코드와 삼각행렬을 단위행렬로 변환하는 코드(그림 29)를 통합하여 최종 해를 구하는 전체 C 프로그램을 그림 33에 보였다.

4. 구현된 가우스 소거법의 비교·분석·검증·개선

4.1 프로그램 시험 방법 및 해의 정확성 비교

최소공배수가 적용된 가우스 소거법의 효과성을 검증하기 위해, 미지수의 계수와 해를 난수 발생에 의한 -16 ~ 15 사이의 정수(단, 0 제외)로 설정하여 연립방정식 10개를 랜덤하게 만들어 최소공배수를 사용하지 않는 경우(그림 31)와 최소공배수를 사용하는 경우(그림 33)에서 구해진 해의 정확도를 분석하도록 한다. 표 1에 시험용 연립방정식과 두 프로그램이 구한 해의 정확도를 요약해 보였는데, 이 표에서 보는 바와 같이 최소공배수가 적용된 방법의 정확도가 훨씬 높다.

4.2 프로그램 자체의 정확성 검증 및 개선

주어진 연립방정식의 특이성에 따른 구현 프로그램의 오류가 없는지를 검증하기 위해서는 충분히 많은 수의 연립방정식에 대한 풀이 결과를 관찰해봐야 한다. 이를 위해 그림 33의 프로그램을 100, 1000, 10000, 100000과 같이 난수 발생에 의한 연립방정식의 개수를 늘려가면서 시험하는 방안을 도입한다. 시험 결과, 'floating exception' 오류를 만나게 되

는데, 이는 가우스 소거 과정에서 계수가 0이 되는 경우에 대한 고려를 하지 못했기 때문이다. 이를테면, 그림 34의 연립방정 식의 삼각행렬을 만드는 과정에서 단계 1을 마쳤을 때,

```
#include <stdio.h>
#define  N       4   /* 4원 1차 방정식 */
int   lcm(int a, int b), gcd(int a, int b);

                    /* 2x + 4y - 2z + 2u = 12 */
                    /* 1x + 3y + 2z - 1u =  9 */
                    /* 3x + 4y - 1z - 2u =  0 */
                    /* 4x + 5y - 3z + 3u = 17 */
void main()
{
  float co[N][N+1]= { 2,   4,  -2,   2,   12,
                      1,   3,   2,  -1,    9,
                      3,   4,  -1,  -2,    0,
                      4,   5,  -3,   3,   17 };
  int   i, j, k, lc, r, r1, r2;

  for (k = 0; k < N-1; k++) {                    계수행렬을
    for (i = k + 1; i < N; i++) {                삼각행렬로
      lc = lcm(co[k][k], co[i][k]);              변환
      r1 = lc /co[k][k]; r2 = lc /co[i][k];
       for (j = k; j < N+1; j++)
          co[i][j] = co[i][j]*r2-co[k][j]*r1;
    }
  }

  for (k = N-1; k >= 0; k--) {
    for (i = k+1; i < N; i++) {                  삼각
      r = co[k][i];                              행렬
      co[k][i] = co[k][i] - co[i][i]*r;          을
      co[k][N] = co[k][N] - co[i][N]*r;          단위
    }                                            행렬
    r = co[k][k];                                로
    co[k][N] = co[k][N] / r;                     변환
    co[k][k] = co[k][k] / r;
  }

  for (i = 0; i < N; i++) {                      단위
     for (j = 0; j < N+1; j++)                   행렬
       printf("%7.2f ", co[i][j]);               및
     printf("Wn");                               해
  }                                              출력
}

int lcm (int a, int b)
{                                                최소
     int  g;                                     공배
     g = gcd(abs(a), abs(b));                    수
     return(a*b/g);                              구함
}

int gcd(int a,  int b)
{                                                최대
    int  big, sml, r;                            공약
                                                 수
    if (a > b) big = a, sml = b;                 구함
    else       big = b, sml = a;
    while (1) {
         r = big % sml;
         if (r == 0) return(sml);
         big = sml, sml = r;
    }
}
```

그림 33. 최소공배수가 적용된 가우스 소거 C 언어 코드
Fig. 33. C code for Gaussian Elimination using LCM

중심식 ②의 y 계수 6과 소거식 ③의 y 계수 0의 최소공배수가 존재하지 않기 때문에 계산이 불가능하다. 즉, 이미 계수가 0으로 되어 있는 경우는 처리할 필요가 없는 것이다. 위 문제를 해결하기 위한 방법으로, 각 단계에서 소거 작업을 하기 전에 소거식들의 소거 미지수의 계수가 0인 것들은 모두 아래로 옮겨 놓고(그림 35에서의 자리바꿈), 이 식들에 대해서는 소거 미지수가 이미 소거된 결과와 동일하므로 소거 작업을 하지 않도록 하는 방안을 적용하였고, 이를 보완한 C 언어 코드를 그림 35에 보였다.

표 2에는 개선된 프로그램을 100~100,000회 생성된 연립방정식에 적용했을 때 정확한 해의 빈도를 측정한 결과를 보였는데, 이 과정에서 오류가 발생하지 않았기

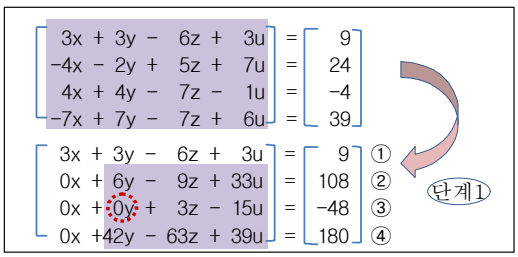

그림 34. 예외 처리가 요구되는 연립방정식의 예
Fig. 34. An Example of Equations Requiring Exception Handling

표 1. 가우스 소거 프로그램들의 해의 정확성
Table 1. Precision of Gaussian Elimination Programs

순	연립방정식 | 해	최소 공배수 미적용 시 해	최소 공배수 적용 시 해
0	-5 11 -14 11 359 | -10 4 6 13 15 -226 | 14 12 12 -15 -9 318 | -15 -15 15 1 14 275 | -5	-10.000000954 13.999999046 -15.000000000 -4.999999046	-10.000000000 14.000000000 -15.000000000 -5.000000000
1	-9 -10 -9 9 -146 | 6 -6 -8 12 9 -145 | 2 -6 6 3 11 -126 | -1 2 -11 -15 8 -67 | -9	6.000000000 2.000000000 -1.000000000 -9.000000000	6.000000000 2.000000000 -1.000000000 -9.000000000
2	7 -2 -11 11 10 | 12 10 -14 9 8 101 | 4 -4 -13 10 6 -56 | 5 -2 15 9 9 72 | -1	11.999999046 4.000000477 5.000000000 -0.999999404	12.000000000 4.000000000 5.000000000 -1.000000000
3	-2 -13 8 -3 -91 | -1 -3 10 14 -10 303 | 15 10 -12 -10 -2 -362 | 15 12 -7 -3 -2 -174 | 6	-0.999996662 15.000000954 15.000003815 6.000002861	-1.000000000 15.000000000 15.000000000 6.000000000
4	11 -1 1 8 16 | 15 -11 -11 -7 12 -318 | 6 3 -3 -6 11 -59 | -15 -8 9 2 3 -144 | -16	15.000000000 6.000000000 -15.000000954 -16.000000000	15.000000000 6.000000000 -15.000000000 -16.000000000
5	-6 5 -14 -10 43 | 2 3 8 -7 2 210 | 13 -15 14 -4 2 222 | -10 -8 1 12 2 -93 | 15	2.000002861 13.000004768 -9.999996185 14.999995232	2.000000000 13.000000000 -10.000000000 15.000000000
6	-1 7 6 9 105 | -10 13 6 10 -8 -138 | 5 4 5 8 12 65 | 1 7 12 -15 11 41 | 6	-9.999996185 4.999999523 0.999999642 6.000000477	-10.000000000 5.000000000 1.000000000 6.000000000
7	9 -9 -15 11 -17 | -3 -1 3 13 1 -21 | 1 -16 13 -11 -3 86 | -2 -3 -5 3 -3 1 | -1	-3.000000477 0.999999523 -2.000000000 -1.000000000	-3.000000000 1.000000000 -2.000000000 -1.000000000
8	-10 -8 4 -10 -396 | 14 10 1 -13 3 209 | 14 1 -8 -12 -9 -212 | -1 11 -6 -16 -16 -138 | 14	13.999997139 13.999999046 -1.000002384 14.000002861	14.000000000 14.000000000 -1.000000000 14.000000000
9	-16 -9 -14 -13 94 | -15 -11 -16 -15 1 -115 | 3 -9 5 -14 -11 17 | 15 -8 -1 14 -2 341 | -7	-15.000000954 3.000002238 15.000000954 -6.999999046	-15.000000000 3.000000000 15.000000000 -7.000000000
정확도		10%	100%

때문에 프로그램 자체의 예외 처리에 대한 검증 테스트가 완료된 것으로 본다.

표 2. 개선된 프로그램의 검증(정확한 해의 빈도)
Table 2. Verification of the Improved Program(Number of the Accurate Answer)

시도횟수 / 유형	100	1000	10000	100000
최소공배수 미적용	9	87	694	7096
최소공배수 적용	98	980	9622	96385

```
#include <stdio.h>
#define N       4
#define NEQA    10000
int lcm(int a, int b), gcd(int a, int b),
random();
void main()
{
  float co[N][N+1], t;
  int   xyzu[N], l, i, j, k, e, lc, r,r1,r2;
  for (l = 0; l < NEQA; l++) {
    for(j = 0; j < N; j++)                    } 해 설정
      xyzu[j] = random();
    for(i = 0; i < N; i++) {                  }
      for(j = r = 0; j < N; j++) {            } 방정
        co[i][j] = random();                  } 식
        r = r + co[i][j]*xyzu[j];             } 설
        printf("%4d ", (int)co[i][j]);        } 정
      }                                       }
      co[i][j] = r;                           }
      printf("%4d | %3dWn", (int)r, xyzu[i]);
    }
    for (k = 0; k < N-1; k++) {               }
      for (e = N-1; e >= k; e--)              } 계수가 0인
        if (co[e][k]) break;                  } 가장 위쪽 수
      if (e < k) continue;                    } 식을 검색
      for (i = k; i < e; i++) {               } 계수 0인 식의
        if (!co[i][k]) {                      } 자리바꿈
          for (j = k; j < N+1; j++)           }
            t=co[i][j],                       }
            co[i][j]=co[e][j],co[e][j]=t;     }
          e--;                                }
        }                                     }
      for (i = k + 1; i < N; i++) {           } 삼각행렬
        if (!co[i][k]) break;                 } 변환
        lc = lcm(co[k][k], co[i][k]);         }
        r1 = lc/co[k][k]; r2 = lc/co[i][k];   }
        for (j = k; j < N+1; j++)             }
          co[i][j]=co[i][j]*r2-co[k][j]*r1;   }
      }                                       }
    }
    for (k = N-1; k >= 0; k--) {              }
      for (i = k+1; i < N; i++) {             } 단위행렬
        r = co[k][i];                         } 변환
        co[k][i] = co[k][i] - co[i][i]*r;     }
        co[k][N] = co[k][N] - co[i][N]*r;     }
      }                                       }
      r = co[k][k];                           }
      co[k][N]=co[k][N]/r;co[k][k]=co[k][k]/r;
    }
    for (i = 0; i < N; i++) {                 }
      for (j = 0; j < N; j++)                 } 해 출력
        printf("%4d ", (int)co[i][j]);        }
      printf("     | %13.9fWn", co[i][j]);    }
    }
    printf("-----------------------------Wn");
  }
}
```

그림 35. 예외 처리된 가우스 소거 C 프로그램
Fig. 35. Gaussian Elimination Program with Exception Handling

IV. 교수학습 자료 적용 및 평가

여기서는 제안된 교수학습 자료를 □□대학교 과학영재원의 2011년 중등 정보수학 프로그램 분야 사사지도에 적용한 내용 및 결과를 분석한다.

1. 사사 지도과정 운영환경

강릉원주대학교 과학영재원의 중등 영재 사사 지도과정의 총 수업 시간은 40시간으로, 주어진 기간 내에 자유롭게 교육하도록 설계되었다. 본 교수학습 자료가 적용된 사사 지도과정은 2011년 3월부터 2011년 8월까지 매월 1~2회 주말을 이용 하여 대면 수업을 하였고, 자가 학습 도중 일어나는 질의응답은 동 과학영재원에서 제공하는 사이버 강의실을 활용하였다.

표 3. 교수학습 자료 강의 시간표
Table 3. Syllabus for the Proposed Instructional Materials

회	내 용	목 표
1	제시된 다양한 형태의 2원 1차 연립방정식에 대한 풀이 과정	가감법, 대입법, 등치법 등은 모두 미지수를 소거하는 과정임을 이해
2,3	가감법, 대입법, 등치법의 C 언어 구현(그림 3~14)	수학문제를 컴퓨터 프로그래밍 언어로 표현한다는 것의 개념 터득
4-1	실생활에서의 예를 중심으로 한 행렬의 기초 표현법	기호나 틀에 의한 수치 관련 상황의 표현의 편리성 및 규칙성 발견
4-2	행렬의 곱셈 규칙을 토대로 한 연립방정식의 행렬 표현	연립방정식을 정형화된 틀에 맞추어 간결하게 표현할 수 있음
5-1	행렬의 곱셈 규칙을 적용하여 양변에 미리 구해진 역행렬을 곱한 결과 생성된 단위행렬 및 해	행렬식으로 표현된 연립방정식의 양변에 적절한 행렬을 곱하면 해를 구할 수 있음을 발견
5-2	주어진 행렬에 대한 역행렬을 구하는 과정	주어진 행렬에 대한 역행렬을 공식으로 쉽게 구함
6	역행렬로 연립방정식의해를 구하는 C 프로그램 구현(그림 22~ 25)	프로그래밍 언어의 배열 사용법 및 편리성 인식
7	삼각행렬을 단위행렬로 변환하여 해를 구하는 과정 및 C 프로그래밍(그림 27~28)	역행렬에 의한 해구하기의 한계 및 보다 일반화된 방법론의 필요성 인식
8	계수행렬을 삼각행렬로 변환시키는 과정 및 C 프로그램 구현, 테스트(그림 30~31)	반복에 의한 문제의 일반화를 위한 창의적 논리 전개 능력 배양
9	계수 일치 과정에서의 정확성 향상을 위한 최소공배수 방법 및 C 프로그래밍(그림 32~33)	정확성 및 프로그램 오류에 관한 창의적 문제 발견 및 해결
10	예외 상황을 극복하기 위한 프로그램 개선(그림 34~35)	문제 발견 및 해결을 위한 예외처리 논리 전개 능력 배양

2. 강의 시간 설계

이 사사지도에 참여한 학생은 지방의 소도시 중학교 3학년 1학기에 재학 중인 2 명의 학생들로서, 학업 성적은 상위권이고, 동 과학영재원의 중등심화 과정에서 C 언어 기초 문법을 이수하였다. 제안된 교수학습 자료는 1일 4시간 기준으로 10회 강의하는 것으로 설계하여 적용하였고, 각 회차별 구체적인 지도 내용과 목표는 표 3과 같다.

3. 학습 성취도 평가

본 논문의 저자들은 제안된 교수학습 자료에 제시된 최종 단계의 가우스 소거 프로그램을 충분히 이해하고 스스로 구현할 수 있는 수준을 컴퓨터 관련 전공 학과의 2학년 과정으로 보고 있다. 질의응답을 통한 두 학생들에 대한 5 단계 평가(매우 우수, 우수, 보통, 미흡, 아주 미흡) 결과를 표 4에 보였는데, 첫 시범 적용에 따른 시행착오를 고려했을 때, 제안된 교수학습 자료가 창의적 문제 발견 및 해결을 위한 하나의 교안으로서 가치가 있는 것으로 판단된다.

표 4. 학생 성취도에 의한 교수학습 자료 평가
Table 4. Evaluation for the Proposed Instructional Materials by the Degree of Students' Achievement

회	학생 A	학생 B
1	매우 우수	매우 우수
2-3	우수	매우 우수
4.1	매우 우수	매우 우수
4.2	매우 우수	매우 우수
5.1	매우 우수	매우 우수
5.2	우수	매우 우수
6	우수	매우 우수
7	우수	매우 우수
8	우수	우수
9	우수	우수
10	보통	우수

V. 결 론

영재 과정에 있는 학생은 물론, 중등 수학 과정에 있는 학생들 대부분이 컴퓨터를 활용하여 수학 문제를 해결한다는 개념 자체에 익숙하지 않다. 이 연구에서는 1차 연립 방정식의 해를 컴퓨터를 활용하여 구하는 과정을 통하여, 수학 문제를 컴퓨터 환경에서 표현하고 그 풀이 과정을 전개하는 하나의 전형적인 교수학습 자료를 스스로 질문하고 스스로 해결안을 모색하는 시나리오로 제시함으로써, 컴퓨터에 대한 새로운 가치를 인식할 수 있는 사례를 제공하였다. 제시된 교수학습 자

료의 코스웍을 마친 학생들은 미지수가 많은 연립 방정식은 컴퓨터의 도움 없이는 풀이가 불가능하다는 점과, 그 과정에서 행렬이라는 표현법의 필요성을 인식하게 된다. 즉, 연립 방정식을 컴퓨터 프로그래밍 언어로 표현하기 위한 수단으로 행렬이라는 틀이 도입되었는데, 이와 같이 수학 영역과 컴퓨터 영역이 만나기 위해서는 표현 기법이라는 가교가 반드시 필요함을 깨닫게 되는 것이다. 또한, 행렬을 다루는 과정에서 고도의 창의력이 요구되는 프로그래밍 언어의 반복(루프)에 의한 논리 전개의 일반화 능력이 크게 향상되기 때문에, 이 교수학습 자료는 컴퓨터 프로그래밍 스킬 향상 측면에서도 매우 유용하게 적용될 수 있다.

참고문헌

[1] Wikipedia Korea(The Free Encyclopedia), http://ko.wikipedia.org/wiki/%EC%B2%9C%EC%9E%AC. Wikipedia Foundation Inc., 2011.

[2] Korea Law Information Center, http://www.law.go.kr/%B9%FD%B7%C9/%BF%B5%C0%E7%B1%B3%C0%B0%C1%F8%C8%EF%B9%FD, Korea Ministry of Government Legislation, 2011.

[3] In Baum, S. M., Reis, S. M., & Maxfield, L. R. (Eds.), "Nurturing the gifts and talents of primary grade students", Mansfield Center, CT: Creative Learning Press, 1998. (http://www.gifted.uconn.edu/sem/semart13.html)

[4] D. E. Knuth, "The Art of Computer Programming, Volume 1-3, Third Edition" Addison-Wesley, 2004.

[5] Jaho Lee, Hyeon-jong Oh, "Design and Validation of Education Contents of Algorithm for the Gifted Elementary Students of Computer Science", Journal of Gifted/Talented Education, Vol. 19, No. 2, pp.353-380, 2009.

[6] Ki-Soon Han, "Current status and future prospect of gofted education programs", The Journal of the Korean Society for the Gifted and Talented, Vol. 5, No. 1, pp.110-129, 2006.

[7] Woo-Cheon Jeon, "An Introduction to Education for the Gifted Children in IT", Dodeulsaekim, 2010.

[8] News paper article, "Gyeonggi Provincial Office Education, Evaluation for the all of the 603 Gifted Education Institutions 603", NewsWin, April 27, 2011. (http://www.newswin.kr/news/articleView.html?idxno=6774)

[9] Mi-Sook Kim, "Instructional Materials for IT Education for the Gifted", Korean Educational Development Institute. December, 2006.

[10] Wikipedia English(The Free Encyclopedia), http://en.wikipedia.org/wiki/Equation, Wikipedia Foundation Inc., 2011.

[11] David M. Burton, "Elementary Number Theory, Seventh Edition", McGraw Hill, pp. 17-31, May 2010.

저 자 소 개

이 형 봉

1984 : 서울대학교 계산통계학과 학사
1986 : 서울대학교 대학원 계산통계학(전산과학)과 석사
2002 : 강원대학교 컴퓨터과학과 박사
1986~1993 : LG전자 컴퓨터연구소 선임연구원
1994~1998 : 한국디지탈(주) 책임컨설턴트(유닉스)
1999~2003 : 호남대학교 조교수
2004~현재 : 강릉원주대학교 부교수
관심분야 : 임베디드시스템, 센서네트워크, 데이터마이닝 알고리즘
Email : hblee@gwnu.ac.kr