韓國 컴퓨터情報學會 論文誌
第11卷 第2號, 2006. 5.

2006-11-2-3-3

초등학생들이 프로그래밍 학습 시 발생하는 오류유형 분석

문 외 식*

Analysis of Error Types occurring on Elementary School Student's Programming Learning

Wae-Shik Moon*

요 약

인지능력이 뛰어난 초등학교 고학년에는 컴퓨터 활용교육 보다 컴퓨터 기초원리 또는 프로그래밍교육이 필요하다. 본 연구에서는 재량·특별활동시간 혹은 방과 후 특기적성시간에 고학년(4,5,6학년)을 대상으로 프로그래밍교육 시 발생 가능한 문제점들을 미리 예측하고 대처하는 방법으로 초등학생들이 프로그램 작성 및 실행과정에서 발생하는 모든 오류들을 수집한 후 이를 유형별로 분류하고 분석하였다. 분석된 오류들을 활용하면 최적의 프로그래밍 교육과정을 작성할 수 있으며 이를 기초로 교수·학습 시 학습효과와 흥미도 유발에 큰 도움을 줄 수 있다. 본 연구에서 수집한 오류들의 분석에서는 초등학생들이 프로그래밍 시 어려워하는 영역들은 소프트웨어 사용미숙으로 인한 단순오류, 영어로 된 예약어 사용미숙 등으로 인한 단순코딩이 가장 많았으며 다음으로 문법 이해의 어려움으로 발생하는 오류들이 가장 많았다. 이러한 오류의 유형들은 상업용 소프트웨어개발 업체에서 분석된 오류의 유형과 반대 현상으로 나타났으나 점차 교수·학습이 향상되면 같은 현상으로 바람직하게 나타날 것으로 예측된다.

Abstract

Higher grade elementary school students who have superior cognitive abilities need education of basic principles of computer or programming rather than computer in education. In this study, all the errors occurring while elementary school students wrote and executed programs were collected, in the method of predicting and dealing with possible-to-occur problems on programming education of the higher grades (4th, 5th and 6th grades) during their optional·special activities or during talent·aptitude activities after school, classified by type and analyzed. If the errors analyzed are put to practical use, optimal programming curriculums could be written and such curriculums could be a great contribution to induction of learning effect and interest on teaching·learning. It was found by analyzing the errors collected for this study that the most of elementary school students during programming felt difficulties in simple errors by poor use of software and in simple coding by poor use of reserved words in English. In the next, students occurred errors by difficulties in understanding grammar. It was exposed that these error types were the opposite phenomena to those analyzed by commercial software developing companies, however, it is predicted that if teaching·learning is getting improved, the same phenomena could be found desirably.

▸ Keyword : programming, optional/special activities, errors, curriculums

• 제1저자 : 문외식
• 접수일 : 2006.03.08, 심사완료일 : 2006.05.20
* 진주교육대학교 컴퓨터교육과 부교수

Ⅰ. 서 론

지식정보 사회가 급격하게 변화함에 따라 정보강국을 자처하는 미국 등 각 국가들은 초·중등 및 대학에서 정보기술교육을 어떻게 가르켜야 될 것인지에 대해 매우 활발하게 연구 중이다. 현재 우리나라의 컴퓨터 및 인터넷 보급률 수준은 세계 최고수준 이지만 단순히 초보적인 응용프로그램(워드프로세서, 파워포인트 등)을 다루거나 오락, 인터넷 검색에만 능할 뿐 교육 및 과학목적으로 사용하는 활용능력은 크게 떨어져 OECD국가 중 평균에 크게 못 미치고 있다.

특히, 컴퓨터의 기본원리를 이해하고 논리적 및 문제해결 능력을 키울 수 있는 프로그래밍 학습을 하는 학생은 전체 한국학생의 8%에 불가해 OECD국가 전체 평균인 23%의 3분의 1에 해당하여 OECD회원국 40개국 중 최하위인 39위로 심각한 컴퓨터활용 왜곡현상이 나타나고 있다[1].

미국, 일본에서는 활용위주의 컴퓨터교육과정에서 개념과 원리를 기초로 하는 교육과정의 비중을 높여 학습자들의 문제해결능력 향상에 초점을 두고 있으며 인도에서는 더 나아가 초등학교 2학년 때부터 로고나 베이직 등의 언어를 이용하여 컴퓨터의 기본원리, 프로그래밍원리를 먼저 배운 후 고학년때 응용소프트웨어를 배우게 하여 어릴 때부터 흥미위주의 컴퓨터학습을 경계하고 창의성을 늘리는데 중점을 두어 IT인력양성을 위한 시발점을 초등학교 때부터 두고 있다.또한, 이스라엘에서는 컴퓨터과학을 컴퓨터교육의 필수교과로 지정하여 기본적 알고리즘 문제와 이론 및 해결방법을 강조하여 프로그래밍 언어교육을 실시하고 있다[2][3]. 그러나 우리나라에서는 기존에 실시하고 있는 컴퓨터교육 정책에서 크게 후퇴하여 초등학교 학교수업에서 컴퓨터교육 자체를 없애고 중등학교에서는 대폭 축소하려는 정책과 대조적이다[4].

초등학교에서의 프로그래밍교육은 초등학생들이 배우기에는 다소 어려운 언어와 문법으로 구성되어 있고 현실적으로 수업시수 및 전문교사 확보가 어렵다.

그럼에도 불구하고 재량활동·특별활동시간 그리고방과 후 특기적성시간 등을 통한 프로그래밍 교육 실시 시도는 초등학교 전체교과 학습 향상에 많은 장점을 가지고 있기 때문에 장점을 흡수하면서 문제점을 피해갈 수 있는 적극적 방안을 모색하면 단순한 컴퓨터 활용 위주 교육의 한계를 뛰어넘어 논리적 사고력 및 문제해결 능력을 향시킬 수 있는 효과적인 IT 교수방법이라 할 수 있다. 본 연구에서는 이러한 필요성을 기초로 특별활동시간에 초등학생 4,5학년(정보영재 기초반 13명)을 대상으로 프로그래밍학습을 시킨 후 주어진 문제를 스스로 알고리즘화 하도록 하고 프로그래밍 하는 과정에서 발생한 오류들을 수집하고 유형별로 정리하여 분석하였다.

분석된 오류자료들을 교사가 활용하면 초등학생들의 프로그래밍교육의 어려운 요소가 어떤 것인지를 파악하여 향후 학습자들이 빈번히 발생시키는 오류들을 제거하는 학습을 시킴으로서 사전에 발생 가능한 오류를 미리 예측하는 능력과 문제해결 능력을 키우고 프로그래밍 학습에 대한 흥미도를 유발하여 지속적인 학습과 성취도 향상에 큰 도움을 줄 수 있다.

Ⅱ. 관련연구

2.1. 컴퓨터교육

컴퓨터교육은 엑셀, 윈도우즈, 워드프로세서 등의 응용소프트웨어 사용을 위한 기본적인 소양교육, 컴퓨터를 활용하여 교과수업의 효과를 높이는 활용교육 그리고 컴퓨터개념 원리 및 프로그래밍학습을 통해 과학적 사고방식의 도입으로 창의성과 문제해결 능력을 높이는 교육이라 할 수 있다.

2.2 초등학교 컴퓨터교육관련 교육과정

7차 초등교육과정은 초등학교 5, 6학년 실과과목의 일부분에서 소양교육을 할 수 있으며 또한, 국어, 영어 교과의 작은 영역에서 소프트웨어 활용학습을 할 수 있는 영역이 일부 있다. 표 1. 의 전체적인 내용을 살펴보면 현재의 초등학생들의 컴퓨터 수행능력을 재대로 반영하지 못한 것으로 대부분 이미 터득한 워드프로세서를 활용하는 간단한 수준으로 학습자들의 흥미도를 잃게하여 컴퓨터교육 자체가 무의미해질 수도 있다. 또한, IT(ICT)의 신기술 수명주기가 1,2년 정도라 본다면 교육과정 자체가 현실성이 매우 결여되어 있다[5][6]. 이러한 현실적 어려움을 해결할 수 있는 방법으로 재량·특별활동시간 등의 교과 활동시간을 통해 1-6학년까지 년 간 34시간을 적절히 활용함으로서 컴퓨터 소양 및 활

용 능력을 크게 높일 수 있으며 방과 후 특기적성시간 등의 비교과활동시간을 활용할 수도 있다. 컴퓨터교육이 정규교과에 반영되어 있지 않기 때문에 초등학교에서 체계적인 교수·학습을 위해서는 재량활동시간에 1, 2, 3(4)학년에는 소양교육 그리고 (4)5, 6학년에는 활용교육과 기본적인 컴퓨터원리 및 프로그래밍 학습이 매우 효과적이다.

표 1. 초등컴퓨터교육 관련 전체 교육과정
Table 1. The entire curriculums related to elementary computer education

과목	학년	영역	컴퓨터 관련 교육내용
국어	3	쓰기	• 글을 컴퓨터에 옮겨 쓴다. - 기본: 글쇠판을 자신이 쓴 글을 컴퓨터에 옮겨 쓴다. - 심화: 글쇠판을 보지 않고 자신이 쓴 글을 컴퓨터로 옮겨 쓴다.
	4	쓰기	• 컴퓨터를 이용하여 자신의 생각을 글로 쓴다. - 기본: 컴퓨터를 이용하여 방학이 되면 하고 싶은 일을 글로 쓴다. - 심화: 컴퓨터를 이용하여 가족신문에 실을만한 내용을 글로 쓴다.
	5	쓰기	• 전달효과를 고려하여 자신의 글을 컴퓨터로 편집한다. - 기본: 글자모양, 크기, 문단모양 등을 고려하여 자신의 글을 보기 좋게 컴퓨터로 편집한다. - 심화: 그림, 사진, 표 등을 넣어 자신이 쓴 안내장이나 신문을 보기좋게 편집한다.
영어	4	읽기	• 심화과정: 컴퓨터 자판의 알파벳 문자 익히기
	5	읽기	• 심화과정: 컴퓨터 자판에서 알파벳 문자를 찾아서 치기
		쓰기	• 심화과정: 컴퓨터 자판에서 구두로
실과	5	생활기술	• 컴퓨터 다루기 - 컴퓨터 구성 - 자판 다루기와 글쓰기
	6	생활기술	• 컴퓨터 활용하기 - 컴퓨터로 그림 그리기 - 컴퓨터 토인 활용하기
재량활동	1 6		• 년 간 34시간 이상 활용 가능 (1학년 30시간, 2-4학년 34시간, 5,6학년 실과교과/특별활동 활용포함) ※초.중등학교정보통신기술운영지침(2000년)

2.3 프로그래밍교육

2.3.1 프로그래밍교육이 미치는 교육적 효과

프로그래밍교육이란 문제해결을 알고리즘으로 설계 및 구현하여 이를 프로그래밍언어라는 도구로 표현할 수 있는 복합적인 능력을 배양하는 교육을 말한다. Van Lengen과 Craig A는 프로그래밍 경험을 통하여 컴퓨터의 처리 과정에 대해 이해하게 되고, 컴퓨터 하드웨어에 대한 추상적인 개념을 확립할 수 있게 된다고 하였으며 Salomon G와 Perkins D.N은 컴퓨터 프로그래밍 영역에서 사용한 전략과 수단은 컴퓨터 영역이 다른 영역에까지도 전이되고 일반적인 문제 해결에 도움이 된다고 하였다[7]. 이와 같은 연구들에 의해 프로그래밍교육이 전통적인 교과에 미치는 교육적 효과는 다음과 같다

첫째로, 프로그래밍 과정을 통해 반복되는 오류의 유형을 쉽게 분석하고 스스로 사고하고, 문제를 해결할 수 있는 자기 반성적이고 논리적인 능력을 키울 수 있어 전통적 교과학습을 자기주도적으로 이끌어 나갈 수 있는 능력을 배양하는 유용한 교육적도구가 될 수 있다.

둘째로, 프로그래밍 학습은 자연적으로 응용소프트웨어가 컴퓨터내부에서 동작하는 과학적인 원리를 이해할 수 있어 하드웨어의 추상적 개념과 과학·수학적인 개념을 자연스럽게 배울 수 있다.

셋째로, 단순한 활용위주의 교육에서 탈피하여 미래 세계 경쟁력을 한 단계 높일 수 있는 IT기술 능력 교육을 초등학교에서부터 시작할 수 있는 계기를 마련하고 정보과학의 영재성을 조기에 발굴할 수 있어 효과의 정도에 따라 국가수준의 컴퓨터교육의 발상과 기본방향을 재 정립할 수 있다.

2.3.2 프로그래밍교육의 교수-학습방법

컴퓨터를 이용하는 교수·학습 형태는 개인교수형(tutor), 보조도구형(tool), 학생주도형(tutee)으로 크게 분류할 수 있다[8]. 개인교수형은 전통적인 교사의 역학을 컴퓨터가 대신하는 형태로 대부분의 CAI 소프트웨어가 이러한 형태에 속하며 현재 각광을 받고 있는 E-learning 또는U-learning 형식을 띄고 있는 학습 방법도 이 형태에 속할 수 있다.

보조도구형은 결과를 얻기 위해 컴퓨터를 도구로서 단순히 사용하는 경우로 흔히 우리가 사용하는 응용소프트웨어를 활용하는 형태를 말한다.

학생주도형은 학생 자신이 직접 구성한 논리적 프로그램 과정을 통해 컴퓨터를 조작하여 실행하도록 하여 결과를 얻

고 스스로의 사고과정을 통해 반성하는 발견적 교수방법 형태로서 BASIC 등의 프로그래밍언어를 이용하여 과정과 결과를 얻는 형태가 될 수 있다. 피아제는 초등학생들의 지적인 성장은 교사의 지도에 의해서 이루어 지는 것이 아니라 아동 자신의 경험이나 사회적 상호작용, 성숙 및 평형 등에 의해서 크게 영향을 받는다고 한다. 따라서, 학생주도형의 프로그래밍교육은 초등학생 자신들이 하고있는 프로그래밍 활동이 가치있고 의미있는 일이라는 것을 스스로 느끼도록 학습환경을 만들 필요가 있다.

프로그래밍 학습 시 발생하는 경험적 문제점들을 정확한 데이터로 제시하기에는 어려우나 일반적으로 목표설정 달성에 미달하는 큰 요인을 두 가지로 정리할 수 있다[8].

첫째, 기능 및 문법 중심으로 교수.학습이 진행되기 때문에 학습시간과 많은 사고력이 소요되는 알고리즘 구현에 소요되는 충분한 시간 배정이 어려워 완성도 있는 프로그래밍 작성에 어려움이 있다.

둘째, 학습자의 개별수준을 무시한 교수.학습 방법으로 학습 중반 이후 학습자의 수업 참여도가 현저히 떨어지는 경향이 있다. 인지능력이 중등에 비해 비교적 떨어지는 초등학생을 위한 프로그래밍 교수·학습 환경을 만들기 위해서는 먼저, 학습자들은 자신의 수준에 맞는 능동적인 프로그래밍 활동을 하고 교수자는 모니터만 해 주고 가능한 손길이 적게 가도록 하는 가변적인 교육과정을 만들 필요가 있다.

즉, 기존의 교육과정과는 특별히 다른 느슨한 교수과정이거나 순차적 교육과정이 아닌 수준에 맞게 선택하여 학습할 수 있는 학습 환경을 만들어야 된다. 교수·학습방법의 유형은 프로그램언어를 처음 배우는 초등학생들에게 언어의 이론적 학습을 위해서는 개인교수형, 그리고 실습을 위해서는 학생주도형과 발견적 교수방법을 함께 적용한 느슨한 교수·학습 방법을 함께 고려한 가변적이고 수준별 교육과정과 교재개발이 필요하다.

2.3.3 프로그래밍교육을 위한 교과영역

초등학교의 교육과정은 전통적인 교과와 특별활동의 두 영역으로 편성되어 있다. 여기에 재량활동을 신설하여 교육과정의 편제를 병렬적으로 3대 영역으로 구성되어있다[9].

전통적인 교과에 비해 특별활동 및 재량활동은 학습자의 필요와 선택, 개성과 형편에 따라 학생이 주체가 되어 융통성 있게 운영할 수 있는 유연한 구조의 교육과정이다. 따라서, 초등학교에서의 컴퓨터교육의 최적 영역으로는 재량·특별활동이라 볼 수 있다.

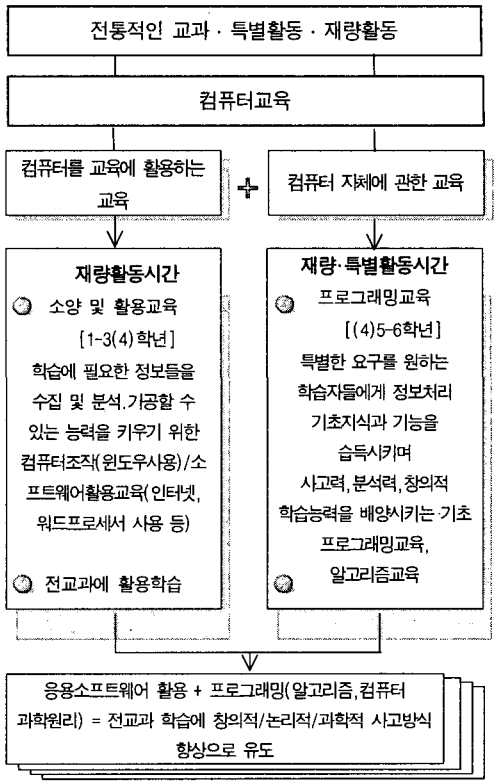


그림 1. 전통적교과, 재량·특별활동, 컴퓨터교육과의 관련구조
Fig 1. The related structure between traditional subjects, optional, special activities and computer education

전통적 교과교육에 활용할 수 있는 ICT소양 및 활용교육은 저학년(1-3학년)들을 대상으로 재량활동시간을 활용하고, 사고력과 인지능력이 가장 뛰어난 4 -6학년을 대상으로 재량활동·특별활동시간에 그리고 비교과영역인 방과 후 특기적성시간에 흥미, 소질, 적성이 비슷한 학생들을 자발적으로 통합 참여시켜 잠재능력과 창의성을 개발시킬 수 있는 기초 프로그래밍 및 알고리즘 교육이 매우 적절하다.

2.4 프로그래밍 오류와 유형

2.4.1 프로그래밍 오류

프로그래밍 오류란 소프트웨어를 개발하기 위한 프로그래밍 전체단계에서 프로그램 작성자의 실수에 의해 만들어진 결함과 최종적으로 만들어진 프로그램이 실제 환경 하에서 사용자가 요구하는 요구명세서 대로 작동하지 않는 모든 현상을 말한다.

즉, 소프트웨어 결함을 발생시키는 우발적 또는 부주의한

행위를 말하며 이는 프로그램 작성 시 연산, 실제값과 불일치, 부정확한 결과를 유도하는 논리 그리고 코딩실수 등을 말한다[11][12].

2.4.2 프로그래밍 오류유형

프로그래밍 오류는 프로그래머의 숙련도 등의 인적사항과 같은 환경적 원인과 프로그램구성과 요구사양의 복잡성 등의 이유로 프로그램 개발 모든 단계에서 발생하며 다음과 같은 유형으로 분류할 수 있다.

2.4.2.1 논리 오류

논리오류는 논리적인 표현 시 부적당한 연산자와 피연산자 사용, 순서에 맞지 않는 논리식 사용, 잘못된 변수 대조, 논리식 또는 조건 테스트 누락, 무한루프와 같은 부정확한 루프 반복횟수를 지정, 중복된 논리지정 등을 들 수 있다.

2.4.2.2 데이터취급 오류

부적절한 데이터 참조 또는 저장오류, 부정확한/부적절한 변수지정, Pack/Unpack오류, 데이터변환오류, 첨자오류 등이 있다.

2.4.2.3 데이터정의 오류

부적절한 데이터 초기화, 부정확한 데이터단위 및 비율, 부정확한 변수선언 등이 있다.

2.4.2.4 인터페이스 오류

화면구성 오류, 잘못되거나 없는 서브루틴 호출, 일관성 없는 서브루틴 호출, 데이터베이스의 부적절한 배치 또는 사용, 인터럽터의 부적절한 취급 등이 있다.

2.4.2.5 설계 오류

요구정의(알고리즘설계)에 적절하지 않는 오류, 불완전하고 일관성이 없는 데이터베이스 및 인터페이스, 예외적인 조건을 무시한 오류, 작업순서에 따른 오류 등이 있다.

2.4.2.6 연산 오류

연산식에서 부적당한 연산자와 피연산자 사용, 부호사용 오류, 부적당한 등식 사용, 복잡한 연산으로 인한 정밀도 저하, 연산식 누락 그리고 자리 반올림 또는 자리 절상(하) 등을 말한다.

2.4.2.7 기타 오류

앞의 오류외에 새로운 소프트웨어 사용 미숙으로 인한 오류, 시스템 하드웨어 오류, 오퍼레이터 조작 오류 등이 있다.

2.4.3 단계별 오류발생 원인

원하는 소프트웨어를 개발하는 전체 과정의 각 단계마다 오류를 발생시키는 요인들이 있다. 일반적으로 오류발생의 약 60%가 자신이 원하는 결과를 도출할 수 있도록 프로그래밍하기 전에 분석하고 설계하는 단계에서 나타나며 약 40%의 오류는 코딩단계에서 발생한다. 이러한 각 단계별 오류발생 원인들을 분석하면 다음과 같다[13].

2.4.3.1 요구정의 단계

프로그래머의 개발능력, 하드웨어 환경을 무시한 무조건적인 요구(시스템환경 무시), 요구사항 누락, 불완전하고 불충분한 상태 요구(불충분한 요구), 참조된 데이터가 일치하지 않거나 혼돈된 정보 및 문서요구(모순된 요구), 참조된 데이터가 일치하지 않은 경우(부정확한 요구) 기타 시스템을 정확하게 파악하지 못하거나 이해력이 부족한 경우 발생한다.

2.4.3.2 설계단계

하드웨어 및 사람능력을 무시한 시스템설계, 불충분하고 구조화되지 않은 설계, 불필요하고 중복된 기능 설계 및 누락된 설계 그리고 기타 표준화되지 않은 언어사용으로 이해하기 어려운 설계 내용 및 예외적인 조건을 누락한 경우에 나타난다.

2.4.3.3 코딩단계

시스템설계에서 없거나 분석되지 않는 부분들을 추가, 프로그램 논리에 맞게 기능을 실행할 수 없도록 부정확하고 모순된 논리를 추가, 프로그래머의 능력부족으로 모듈기능 및 데이터 정의 이해부족, 불필요한 논리추가, 프로그래머의 타이핑 오류, 설계된 논리대로 코딩하지 않고 프로그래머의 판단에 의한 코딩 그리고 기타 구조적 결함 또는 프로그램 언어에 대한 충분한 지식 부족 등에 의해 나타난다.

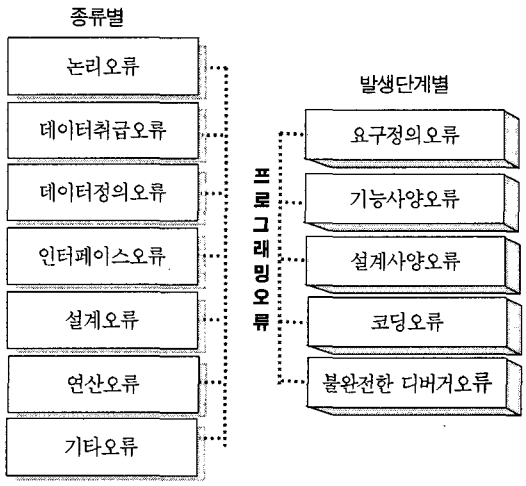


그림 2. 프로그래밍 오류 종류
Fig 2. Kinds of programming errors

III. 오류데이터 수집 및 분석

3.1 데이터수집 대상 및 방법

본 연구에 필요한 오류데이터 수집은 경상대학교와 진주교육대학교에서 공동 운영하는 과학영재 교육원의 원생 중 기초정보영재교육을 받는 13명의 초등학생(4,5학년생)들을 대상으로 4주간(주당 3시간) 기초 프로그래밍(컴파일 및 실행/인터페이스 설계를 위한 기본 컨트롤/변수 및 데이터형/연산자/제어구조)학습 후 2006년 1월 16일 부터 - 20일까지 6과제의 프로그램을 작성하게 하였다. 이때, 프로그래밍 작성부터 실행과정까지 발생한 모든 오류들을 4명의 오류검사원에 의해 수집하게 하였다. 사용된 프로그래밍 도구는 초등학생들이 이해하기 쉽고 친숙한 비주얼베이직 6을 사용하였다.

실험에 사용된 프로그래밍 과제는 초등수학 5-가의 배수와 약수, 분수의 곱셈 그리고 약분과 통분 영역을 응용하여 과제를 해결하는 알고리즘 구현과 프로그래밍 문제이다.

3.2 오류데이터 분류 및 수집

수집된 오류의 분류는 기존에 연구 발표한 Lipow[14]의 오류분류(8항목)를 참조하여 실험 대상의 초등학생들이 빈번히 발생시키는 오류항목들을 4항목(논리오류, 데이터정의 및 취급오류, 인터페이스 오류, 기타오류)으로 구분하여 재분류하였다. 또한, 각 항목별로 세부적인 오류의 종류는 표 2. 처럼 모두 16종류로 구분하였다. 수집한 오류의 총 건수는 13명의 초등학생들이 6과제의 프로그래밍을 수행하는 과정에서 발생한 총 오류수를 누적한 수 이다.

표 2. 오류 데이터
Table 2. Error data

항 목	종 류	오류 건수
논리오류	부적절한 연산자 사용	0
	잘못된 계산식	0
	부적절한 논리개념 도입	6
	논리식 오류	12
	무한루프	0
데이터정의 및 취급오류	부적절한 변수 지정	20
	부적당한 제어구조 사용	2
	잘못된 문법사용	15
인터페이스오류	화면설계 오류	6
	부적절한 객체 배치 및 속성 잘못지정	17
	입.출력 지정 오류	0
기타오류	프로그램 사용 미숙으로 인한 오류(열기/저장 등)	18
	단순한 오타	26
	하드웨어 장애로 인한 오류	2

3.3 오류데이터 분석

3.1.1 항목별 오류 발생

오류의 항목별 전체 분포비율은 논리오류가 14.52%이고 데이터정의 및 취급오류가 29.84%, 인터페이스오류가 18.55% 그리고 기타오류가 가장 높은 비율인 37.09%로 나타났다. 이것은 소프트웨어 개발업체에서 소프트웨어 개발과정에서 일반적으로 가장 높은 비율로 나타나는 논리오류 비율과 대조적인 결과를 나타내고 있다.

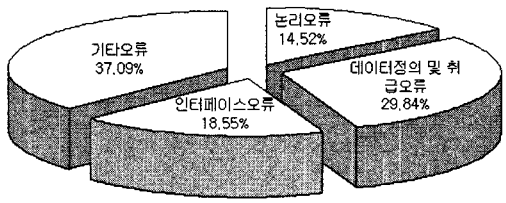


그림 3. 항목별오류 발생 분포비율
Fig 3. Distribution rates by error items

3.1.2 종류별 오류 발생

논리오류에서 종류별 오류 발생 경향은 부적절한 논리개념 도입이 4.0% 이고 논리식 오류가 9.67%로 나타났다. 이것은 실제 프로그램작성 과정이 소프트웨어 개발업체처럼 소프트웨어 규모가 크거나 복잡한 논리적 개념을 요구하는 프로그래밍 과정이 아니었기 때문에 비교적 낮게 나타난 것으로 판단된다. 향후 프로그램 복잡도가 높아지면 가장 많은 오류가 발생할 것으로 판단된다.

데이터정의 및 취급오류에서의 종류별 오류 발생 분포는 부적절한 변수 선언 및 지정이 16.12%, 부적당한 제어구조 사용이 1.61%, 잘못된 문법사용이 12.09%로 초등학생들이 프로그램작성에서 어려운 영역으로 판단되어 향후 프로그래밍 학습 시 프로그래밍언어 문법학습에 특별한 관리가 필요하다.

인터페이스오류의 종류별 오류 발생 경향에서는 화면설계 오류가 4.83%, 부적절한 객체 배치 및 속성 잘못지정이 13.70%로 초등학생들에게 프로그래밍 학습 시 인터페이스 기본개념 교육에 교수·학습시간을 많이 할애해야 할 것으로 판단된다. 기타오류에서의 종류별 오류발생 분포는 열기 또는 저장 등의 프로그램 사용 미숙으로 인한 오류발생이 14.51%로 나타났으며 단순한 오타가 20.96%로 가장 많은 오류로 나타났다. 이는 코딩이 영어로 되어 있고 코딩 자체가 다소 생소해 숙련되지 않은 관계로 나타나는 현상으로 일정한 시간이 경과하면 오류를 가장 많이 줄일 수 있을 것으로 판단된다.

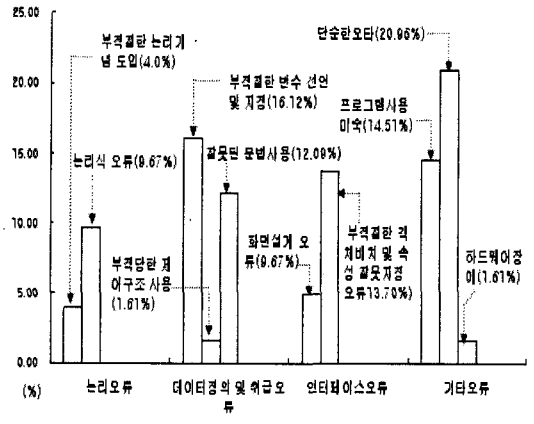


그림 4. 종류별오류 분포비율
Fig 4. Rates by kinds of errors

3.4 선행 연구된 오류데이터와 비교

3.4.1 Lipow, Pointer, Motley의 평균 오류데이터

Lipow, Pointer 그리고 Motley와 Brooks가 전문적인 프로그래머들로 부터 수집 분류한 오류의 평균 발생빈도 연구결과는 표 3. 과 같다[10][14].

표 3. Lipow 등의 평균 오류데이터 발생빈도
Table 3. Average incidence of error data such as Lipow

오류종류	평균 발생빈도(%)
논리오류	30
데이터취급 오류	17
데이터정의 오류	7
데이터입.출력 오류	11
인터페이스 오류	17
데이터베이스 오류	4
연산 오류	8
기타 오류	5

3.4.2 신입 프로그래머의 경험적 오류데이터

창원·마산지역 10인 이상 고용인원을 가진 50개의 기업체 전산실 및 중소 소프트웨어 개발업체에서 프로그래밍 경력이 1년 전·후인 신입 프로그램머가 소프트웨어 개발 과정에서 경험적으로 느낀 발생 오류들을 설문형식으로 조사한 오류데이터 발생빈도는 표 4. 와 같다[10].

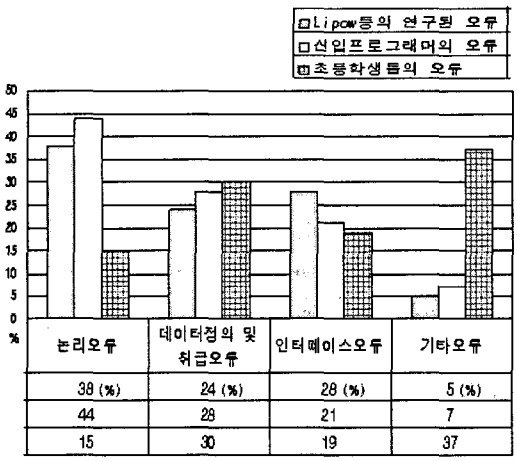
표 4. 신입 프로그래머의 경험적 오류데이터 발생빈도
Table 4. Empirical incidence of error data by new programmers

오류종류	평균 발생빈도(%)
논리오류	31
데이터취급 오류	13
데이터정의 오류	15
데이터입.출력 오류	10
인터페이스 오류	11
데이터베이스 오류	7
연산 오류	13
기타 오류	7

3.4.3 기존 연구결과와 본 연구 오류데이터 비교

기존 연구결과의 평균 오류데이터들과 본 연구에서 초등학생들이 프로그래밍 과정에서 발생했던 오류데이터들의 분포도의 비교는 그림 7. 과 같다. 그림 7. 은 기존 연구 결과에서 나열한 오류항목들 중 유사한 항목들을 서로 묶어 초등학생들에게서 수집한 오류항목들과 일치시켰다.

이때, 함께 묶은 항목들 중 데이터 취급 및 정의 오류를 하나로 묶었으며 데이터 입·출력 오류와 인터페이스 오류를 함께 묶어 인터페이스 오류로 그리고 연산오류를 논리오류에 포함시켜 분류하였으며 데이터베이스 오류항목은 본 연구의 비교 대상인 항목과 다소 거리가 있고 기존 연구에서도 평균비율이 가장 낮아 무시하였다.



	논리오류	데이터정의 및 취급오류	인터페이스오류	기타오류
	38 (%)	24 (%)	28 (%)	5 (%)
	44	28	21	7
	15	30	19	37

그림 5. 기존 연구된 오류와 초등학생들의 항목별 오류 비교
Fig 5. Comparison between the existing errorsand elementary school student's errors

논리오류 발생 유형에서는 전문소프트웨어를 개발하는 업체의 프로그래머들에게는 40% 전후의 가장 높은 비율의 오류발생이 나타나는 것으로 조사되었지만 초등학생들에게는 정반대로 가장 낮은 오류발생 비율로 나타났다. 이는 초등학생들의 프로그램 평가과제가 소프트웨어 개발업체와 같이 규모가 크고 복잡한 프로그램 작성을 요구한 것이 아닌 가장 큰 이유가 된다.

데이터정의 및 취급오류 발생 유형에서는 기존 연구 및 초등학생들 모두에게서 24~30% 까지의 비교적 높은 비율의 오류발생이 나타났다.

기존 연구에서는 프로그램의 복잡도와 사용자의 편리성이 요구되는 상업용 소프트웨어 개발을 위해서는 설계단계가 가장 중요한데 이러한 설계단계가 치밀하지 않은 까닭으로 인해 나중에 코딩과정에서 배열, 부정확한 변수정의 등의 문제로 발생할 수 있을 것으로 판단되며 프로그램 작성 경력이 높은 집단이 그렇지 않은 집단에 비해 낮은 것으로 나타났다.. 그러나 본연구의 실험집단인 초등학생들 대부분이 프로그래밍 언어의 문법적 이해부족에서 많은 오류(30%)를 발생시키는 것으로 판단된다. 인터페이스 오류 발생 유형에서는 프로그램 경력이 높고 전문적 일수록 오류발생 정도(28~21%)가 높게 나타났다. 이는 사용자의 요구도를 충족할 수 있는 비교적 복잡한 소프트웨어 개발에 참여하는 프로그래머들의 집단이 높은 경력의 소유자일 것으로 판단되고 신입 프로그래머들에게는 상대적으로 단순한 인터페이스를 요구하는 프로그래밍에 참여하는 경향이 많은 것으로 추측된다. 초등학생들이 가장 낮은 비율(19%)의 인터페이스 오류 발생은 실험 과제 프로그램이 단순한 논리적 문제를 해결하는 수준의 프로그램이었기 때문이다.

기타오류 발생 유형에서는 단순한 오타, 프로그램 사용미숙 등으로 인한 오류는 신입 프로그래머 또는 전문 프로그래머에게서는 가장 낮은 오류발생 비율(5~7%)로 나타났다. 그러나 초등학생들에게는 코딩자체의 생소함 그리고 코딩이 영문으로 해야 하기 때문에 단순오타 그리고 소프트웨어 작동 미숙으로 인한 오류가 전체 오류의 37%로 대단히 많다. 이는 지속적인 프로그램 작성과정과 학습을 통해서 자연스럽게 해결할 수 있는 단순문제인 것으로 판단된다.

Ⅳ. 결론

초등학교에서의 프로그래밍교육은 전체교과 특히, 수학 및 과학의 논리적 문제 해결 능력 향상에 많은 영향을 줄 수 있다. 본 연구에서는 인지 능력이 뛰어난 초등학생 4-6 학년들이 교과와 연관하여 프로그래밍 학습 시 어떤 문제점과 어려움이 발생할 수 있는지를 파악하여 적절히 대처하기 위한 방법으로 오류데이터를 수집하고 유형별로 분류하고 분석하였다.

이러한 분석데이터를 피드백 시켜 활용하면 교수자가 학습자들인 초등학생들에게 프로그래밍학습 시작 전에 미리 학습 난이도를 예측하여 최적 교육과정을 작성하여 학습시킴으로서 학습자들의 흥미도를 높여 프로그래밍 학습 초기부터 완료단계까지 학습 성취도를 높일 수 있다.

실제로 6과제의 전체 프로그램 완성도 및 성취도는 프로그램 수행과정 중에 실시한 자기만족도 설문조사에서 87%의 높은 결과로 나타났다.

분석된 오류데이터에서 가장 오류건수가 많은 기타오류에서 프로그래밍 기초학습에서 기본적인 코딩방법 및 실습을 통해 초등학생들에게 다소 생소한 영문코딩에 익숙 시킬 필요가 있으며 또한, 프로그래밍 작동을 위한 소프트웨어 사용방법 학습에 다소 충분한 시간을 늘릴 필요가 있다. 두 번째로 많은 데이터정의 및 취급오류의 문제점은 변수, 제어구조, 배열 등의 프로그래밍언어의 문법 이해가 초등학생들에게 큰 어려움으로 나타났다. 실제로 이 영역에 가장 많은 시간을 할당하여 교수·학습할 필요가 있으며 초등학생들이 이해도와 활용능력 배양에 중요한 포인트가 된다.

약 79% 정도인 기타오류, 데이터정의 및 취급오류를 줄이는데 초점을 두면 성공적인 프로그래밍 교수·학습이 될 것으로 판단되며 이러한 종류의 오류가 점차 줄게되면 기존 연구에서 처럼 논리오류와 인터페이스 오류 발생비율이 상대적으로 가장 많이 나타날 것으로 예측된다.

초등학교 프로그래밍 교육은 인지능력이 필요한 영역이므로 모든 초등학생들에게 반드시 교수·학습할 필요는 없다. 그러나 적응 가능한 일부 고학년생 또는 정보영재 학생들에게는 우수한 도구로 판단되며 이를 활용하면 전통적인 교과학습을 향상시킬 수 있는 새로운 교육적 패러다임이 될 수 있다.

참고문헌

〔1〕 한국교육과정평가원, "ICT활용과 관련한 학업성취도 국제비교", 보고서, 2006년 1월 24일.

〔2〕 김현철, "외국의 컴퓨터교육과정 사례를 통한 지식정보 교육과정 분석", 한국정보교육학회(워크샵), 2005년 8월.

〔3〕 정재열외 2, "한·일인도 컴퓨터교육과정의 비교 및 문제점 제시를 통한 우리교육과정의 개선방안", 한국컴퓨터교육학회지 제1권제1호 통권제1호, 2005년.

〔4〕 한태명, "ICT교육의 현황과 컴퓨터교육", 한국정보처리학회/한국정보과학회/한국정보교육학회외 2 공동 주최 (IT 인력 양성과 컴퓨터교육 워크샵), 2005년 7월.

〔5〕 교육인적자원부, "교육정보화, 2001년 교육정보화촉진 시행계획", 2001년.

〔6〕 문외식, "초등학교 ICT활용을 위한 컴퓨터교육과정 모델", 한국교육과정평가연구 논문집 제5권 제1호, 2002년.

〔7〕 이미숙, "컴퓨터기능교육에서 초인지를 이용한 협력적 성찰수업 모형의 개발 및 적용", 한국정보교육학회, 제9권제 4호, 2005년.

〔8〕 김미량, "컴퓨터 프로그래밍교육에 적용 가능한 효과적 교수방법의 탐색적 대안", 한국컴퓨터교육학회 논문지 제5권 제3호, 2002년.

〔9〕 교육인적자원부, "특별활동지도서(계발활동)", 교원대학교 특별활동 국정도서편찬위원회 p26. 53, 2003년.

〔10〕 문외식, "소프트웨어 개발과정에서 발생할 수 있는 오류유형 분석", 진주교대 논문집 제41집, 1999년.

〔11〕 김경희, "소프트웨어 아키텍처의 성숙평가 모델에 관한 연구", 한국컴퓨터정보학회 논문집 제10권 6호", p.168-169, 2005년. 12월

〔12〕 Muneo Takahssi and Yuki Kamyachi, "An Empirical Study of A Model for Program Error Prediction", Proceedings of 8th IEEE Conference on Software Engineering", pp. 331-339, 1985.

〔13〕 Georage rawvski, "Identification of Factors which Cause Software Failure", Proceedings Annual Reliability and Maintainability IEEE Symp, 1992.

〔14〕 Myron Lipow, "Prediction of Software Failure", Journ of System and Software, Vol 1 No 1, 1979.

저 자 소 개

문 외 식

1980년 울산대학교 전자계산학과(공학사)
1986년 부산대학교 전산학전공(공학석사)
1996년 경남대학교 컴퓨터공학과(공학박사)
1981-1984 한국전력공사 전자계산소 근무(4급)
1985-1997 창원전문대학 전자계산과 교수
1998-진주교육대학교 컴퓨터교육과 부교수
〈관심분야〉 ICT활용교육, 컴퓨터교육과정, 소프트웨어 평가, 프로그래밍 및 알고리즘교육