Journal of Digital Convergence
Vol. 17. No. 3, pp. 25-33, 2019

ISSN 1738-1916
https://doi.org/10.14400/JDC.2019.17.3.025

단계적 블라인드 프로그래밍 실습과정을 적용한 소프트웨어 기초교육에 관한 연구

정혜욱
경기대학교 융합교양대학 교양학부 조교수

A study on basic software education applying a step-by-step blinded programming practice

Hye-Wuk Jung
Assistant Professor, Division of Information Communication, College of Liberal Arts and Interdisciplinary Studies, Kyonggi University

요 약 최근 대학에서는 4차 산업혁명 시대에 활약할 수 있게 소프트웨어 기초교육을 강화하고 있다. 비전공 학생들의 경우 프로그래밍에 대한 기본지식이나 전공과목과의 연계성이 낮기 때문에 이들의 이해를 돕기 위한 다양한 교수법이 필요하다. 따라서 본 논문에서는 비전공자를 대상으로 하는 소프트웨어 기초교육의 개선 방안을 제언하고자 한다. 이를 위해, 시연중심모델을 기반으로 단계적 블라인드 처리된 프로그래밍 실습과정을 적용한 학습모델을 설계하여 실제 수업에 적용하고, 학습자들의 문제해결 능력을 분석한 결과, 주차가 진행될수록 학습자 스스로 문제를 해결하는 비율이 상승되는 것을 확인하였다. 제안하는 방법을 통해, 비전공자에게 프로그래밍 과목에 친숙해질 수 있는 기회를 제공하고 지속적인 학습동기를 부여할 수 있다. 후속 연구에서는 보다 다양한 측면에서 학습자들의 학습성과를 분석하고 학습내용의 난이도에 따른 효율적인 교수법에 대한 연구가 필요할 것으로 사료된다.

주제어 : 소프트웨어 기초교육, 소프트웨어 학습모델, 컴퓨터 프로그래밍, 비전공자, 소프트웨어 융합

Abstract Recently, universities have been strengthening software basic education to be active in the era of the fourth industrial revolution. Non-majored students need a variety of teaching methods because they have low knowledge of programming or a lack of connectivity with major courses. Therefore, in this paper, a learning model applying the step-by-step blind programming practice based on the Demonstration Modeling Making model was designed and applied to the actual lecture. As a result of analyzing the problem solving ability of the learner, it was confirmed that the learner's self - solving ratio increased as parking progressed. In the following study, it is necessary to analyze the learner's learning results in various aspects and to study effective teaching methods according to the difficulty of the learning contents.

Key Words : Software Basic Education, Software Learning Model, Computer Programming, Non-Majors, Software Convergence

*This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2015R1D1A1A01061064)
*Corresponding Author : Hye-Wuk Jung(wukj@kyonggi.ac.kr)
Received January 14, 2019 Revised February 26, 2019
Accepted March 20, 2019 Published March 28, 2019

1. 서론

현재 IT융복합 시스템이 다양하게 발전되면서 기술개발을 위한 아이디어 도출의 다양성이 중요하게 고려되고 있다. 따라서 인문·사회 등 다양한 학문의 전공자들의 상상력과 소통력을 IT융복합기술에 접목할 수 있게 교육시키는 것에 대한 관심이 증가하고 있다[1].

최근 대학에서는 변화하는 시대에 적응할 수 있는 인력양성을 위한 소프트웨어 교육 관련 교과목 개설 및 커리큘럼을 도입하여 전교생 또는 비전공 학생들에게 소프트웨어 기초교육을 시키고 있다. 특히, IT를 전공하지 않는 대학 신입생의 경우 소프트웨어에 대해 경험 할 수 있는 기회가 부족한 실정이기 때문에 대학에서 비전공자를 대상으로 하는 소프트웨어 교육은 빠르게 확산되고 있다.

프로그래밍 과목은 주차별 학습내용이 진행되며 계속적으로 누적, 반복되어 새로운 내용과 결합 및 확장되는 특성이 있으므로 충분한 복습과 연습이 필요하다. 그러나 소프트웨어를 처음 접해본 비전공 신입생의 경우 주차별 학습내용에 대해 스스로 복습하고 연습하는 것에 대하여 부담감과 어려움을 느끼고 있다. 또한, 컴퓨터 프로그래밍의 기본원리를 배우지 않은 상태에서는 알고리즘 작성의 어려움을 많이 느끼게 된다. 예를 들어, '양의 정수를 누적하는 프로그램의 알고리즘을 작성해라' 라는 문제를 준 후 알고리즘을 작성하는 경우 변수의 개념이나 반복문의 원리를 모르는 상태로 작성하게 되면 프로그램에 적용하기 어려운 알고리즘을 작성하게 된다. 따라서 프로그램의 기본원리를 충분히 익히고 해당 문제를 다양하게 접해본 후 알고리즘 및 프로그래밍 작성을 하는 것이 바람직하다.

이러한 프로그램 수업은 실습이 중요하기 때문에 일반적으로 예제를 작성해보거나 주어진 문제에 대해 프로그래밍을 하고 결과를 도출하는 방식으로 진행된다. 그러나 프로그래밍 과목을 교양과목으로 수강하는 비전공 신입생의 경우 프로그래밍에 대한 기본지식이나 전공과목과의 연계성이 낮기 때문에 학생들의 이해를 돕기 위한 보다 다양한 교수법이 필요하다.

이에 본 논문에서는 소프트웨어 기초교육을 위해 단계적 블라인드 처리된 프로그래밍 실습과정을 적용한 학습모델을 설계하고, 이를 실제 수업에 적용한 후 학습자의 문제해결 능력 변화에 대한 분석과 학기말에 수행한 프로젝트 내용 및 소감문에 대해 정성적으로 분석한다. 그리고 이를 기반으로 비전공자를 대상으로 하는 소프트웨어 기초 교육의 개선방안을 제언하고자 한다.

2. 이론적 배경

2.1 비전공자를 위한 컴퓨터 기초교육 과정

소프트웨어와 관련된 국내·외 컴퓨터 기초교육 과정을 살펴보면 다음과 같다.

국외대학의 경우 비전공자를 대상으로 컴퓨터 이론 및 프로그래밍 기법을 가르치고 있다. 하버드대학, 예일대학에서는 교양 선택과목으로 컴퓨터과학 입문과정 'CS50'을 개설하여 비전공 학생들에게 컴퓨터원리, 기초 알고리즘, 기초 프로그래밍 등을 가르친다. 이 수업은 학생들에게 컴퓨터에 흥미를 가지며 기초적인 프로그래밍 지식을 습득하게 하고 프로그래밍 능력과 컴퓨터에 대한 이해가 중요함을 인지하게 하여 진로 선택에 영향을 받게 하고 있다[1,2]. 컬럼비아대학은 비전공 학생들을 대상으로 'Computing in Context' 과정을 개설하여 프로그래밍 언어와 컴퓨팅의 개념을 자신의 전공분야에 적용하는 방법에 대해 배울 수 있게 한다. 6~7주 동안 학습한 Python 언어를 이용하여 프로그래밍과 사회과학 컴퓨팅, 경제 및 금융 컴퓨팅, 디지털 인문학 등과 관련된 프로젝트를 수행하는 과정도 포함되어 있다[3].

국내 대학에서는 비전공자들의 컴퓨터교육을 위하여 소프트웨어 관련 교과목을 개설하고 기초적인 프로그래밍 교육을 실시하고 있다. 성균관대학교는 2016년도 신입생부터 소프트웨어 기초교과목으로 '컴퓨팅 사고와 SW코딩', '문제해결과 알고리즘' 과목을 개설하고 교양필수로 지정하였다. 컴퓨팅 사고와 SW코딩 과목은 엔트리와 파이썬을 활용하여 문제해결의 절차 설계 및 구현하는 방법을 다룬다. 문제해결과 알고리즘 과목에서는 논리적 사고력 향상과 소프트웨어 사용능력을 키우기 위하여 프로그래밍 언어로 파이썬을 사용하고 피지컬 컴퓨팅을 실습을 위해 햄스터를 활용한다[4]. 아주대학교는 2015년부터 비전공자를 대상으로 SW 기초교육과정을 개설하여 컴퓨팅 사고, 데이터분석, 프로그래밍 분야를 교육하고 있다. 교양 선택 과목은 프로그래밍기초 교과목에서는 C언어를 이용한 문제분석 및 설계방법과 기초 프로그래밍 능력배양을 목표로 하고 있다. 또한, 인문사회 데이터 분석 연계전공이나 ICT융합 전공자를 대상으

로 프로그래밍언어 강좌를 개설하여 소프트웨어를 다양하게 학습할 수 있는 기회를 제공하고 있다[5]. 이밖에 2015년부터 소프트웨어 중심대학으로 선정된 가천대, 고려대, 부산대 등에서는 각 대학 특성에 맞게 비전공자를 위한 소프트웨어 기초교육과정을 다양하게 개설하여 컴퓨팅 사고에 기반을 둔 문제 해결 능력 및 프로그래밍 교육을 하고 있다[6,7].

2.2 비전공자의 소프트웨어 교육

대학에서는 비전공자들의 소프트웨어 기초교육을 위해 프로그래밍 언어로 Python, C, 앱인벤터 등 전공자들이 배우는 언어를 사용하고 있다. 이러한 언어들을 비전공자들에게 이해시키기 위하여 다양한 교육방법이 연구되고 있다.

피수영(2016)은 비전공자들의 컴퓨팅 사고력 및 문제해결능력의 향상을 위한 방법으로 자기 주도적으로 학습할 수 있게 하는 플립러닝(Flipped Learning)을 이용한 수업진행 방법과 경진대회를 통한 독창적인 앱의 개발과정을 제안하였다[8].

박성희(2016)는 앱개발 또는 스크래치 프로그램을 활용하는 수업에서 교수자의 역할인 교과과정 통합, 프로젝트 행동계획과 관리, 전체 프로젝트 공개행사 항목이 단계적으로 설계된 교수학습모형을 제시하였다[9].

이승현 외(2017)는 성균SW교육원(SSEN)에서 비전공자를 대상으로 시행 중인 온·오프라인 교과목과 이론교육 및 하드웨어와 오픈소스를 활용한 실습 교육과정을 소개하고 정규수업에 적용 한 결과, 학습자들의 문제 해결 능력과 창의력이 향상되었다고 보고하였다[10].

이영석(2018)은 비전공자를 위한 파이썬 기반 소프트웨어 교육 모델, 학습 절차, 커리큘럼을 제안하고 실제 교양수업에 적용하여 컴퓨팅 사고력 향상에 효과가 있음을 확인할 수 있는 결과를 도출하였다[11].

이와 같이 비전공자를 대상으로 하는 교육에 대한 기존 연구들은 대부분 개발한 교과과정을 통한 학습과 프로젝트 경험을 통해 컴퓨팅 사고력 및 창의력을 키울 수 있게 하는 방법에 대해 제안하고 학습자들의 설문조사를 통해 그 효과를 검증하였다. 그러나 소프트웨어에 대한 경험과 기초지식이 부족한 비전공 신입생들이 보다 프로그래밍 작업에 익숙해 질 수 있게 하기 위해서는 수업시간 동안 프로그래밍 기법에 적응하는 과정을 관찰하며 실습시간을 활용하여 프로그래밍 작업에 익숙해 질 수 있게 하는 방법에 대한 연구가 필요하다. 이에 본 연구에서는 실습 시간을 통해 학습자 스스로 충분히 문제해결을 위한 탐구할 수 있도록 블라인드 처리된 응용문제를 해결하는 과정을 적용하여 학습모델을 설계 후 실제 수업에 적용하고 그 효과를 분석하였다.

3. 제안하는 학습모델

3.1 학습모델

본 연구에서는 한국교육개발원에서 제시한 5가지 CT(Computational Thinking) 학습모델 중 시연-모방-제작 단계로 구성되는 시연중심(DMM, Demonstration Modeling Making)모델의 교수학습 절차를 참고하였다[12]. 이 모델은 프로그래밍의 기초적인 문법이나 명령어의 사용방법을 교수자의 시연을 통해 학습자들이 학습한 후 모방 및 연습을 반복하며 프로그램을 제작해보는 과정으로 구성되어 있다. 따라서 이러한 교수법은 프로그램을 처음 접하는 학습자가 프로그래밍 기능 및 작성 방법에 익숙해 질 수 있도록 반복적으로 연습할 수 있는 형태이므로, 본 연구에서는 이 모델에 기초하여 학습모델을 설계하였다.

제안하는 학습모델에서는 Fig. 1과 같이 1주~15주까지 매주 수업시간 내용을 “이론학습 및 예제연습 → 응용문제 실습 → 프로젝트” 과정으로 구성하였다.

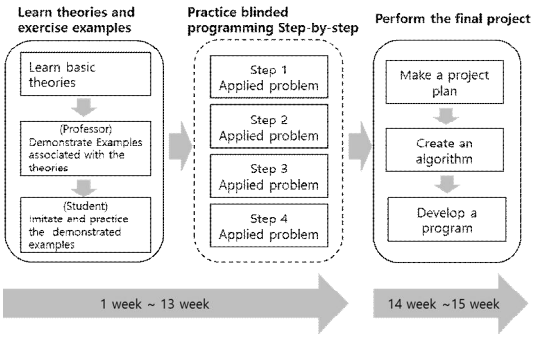

Fig. 1. Learning model

매주 수업 시간은 이론 설명과 실습으로 나누어 진행된다. 수업 초반에는 DMM모델에서 제시한 방법으로 해당 주차의 단원에 대한 이론적인 내용을 설명 후 교수자가 예제를 시연하고 학습자는 시연내용을 모방하여 문제

를 연습해본다. 수업 중반부터는 본 논문에서 제안하는 방법인 단계적으로 블라인드 처리된 응용문제를 풀며 프로그래밍 실습을 한다. 학기말에는 기말 프로젝트 수행을 위하여 프로젝트에 대하여 계획하고 알고리즘 작성 및 프로그램을 개발한다.

3.2 단계적 블라인드 프로그래밍 방법

응용문제 실습에서 적용되는 단계적 블라인드 프로그래밍 방법은 완성된 프로그램의 일부를 부분적으로 은닉하거나 전체소스 또는 결과 값을 은닉한 상태로 프로그래밍 연습을 하는 것으로 정의한다.

학습자들은 단원별 학습내용이 진행되면서 새롭게 배우는 다수 개의 프로그래밍 기법을 암기방식으로 모두 기억하기에는 한계가 있다. 따라서 단계적으로 블라인드 처리된 문제를 통해서 앞에서 배운 내용을 스스로 재확인하는 실시간 복습과정을 유도할 수 있다. 이러한 과정을 매주 수업시간에 반복하여 진행하게 되면 한 학기동안 누적되는 다양하고 많은 원리들을 지속적으로 되뇌는 훈련을 하게 된다.

단계적 블라인드 프로그래밍 실습은 Table 1과 같이 응용문제 1단계~4단계까지로 구성하였다. 각 단계에서 공개 또는 블라인드 처리되는 항목은 일부 소스코드, 추가조건 소스코드, 전체 소스코드, 결과 값이다.

Table 1. Practice contents of blinded programming step-by-step

Step	Goal	Blinded Contents
1	To complete a partially blinded source code	Partial source code
2	To modify a completed source code by adding a condition	Source code of additional condition
3	To write a source code for a problem with given result values	Total source code
4	To write a source code based on a given algorithm	Total source code, Result values

1단계에서는 주차별 학습내용에 해당하는 은닉된 내용의 일부분을 완성하며 프로그램의 흐름을 익힐 수 있게 한다. 이때, 블라인드 처리된 일부 소스코드는 해당주차에 학습한 내용이다. 2단계에서는 완성된 소스코드에 추가 또는 변경 조건을 적용하여 프로그램을 수정해봄으로써 프로그램의 재구성 원리를 학습한다. 3단계에서는 실습 문제의 결과 값만 주어진 상태에서 소스코드를 작성해본다. 4단계에서는 주어진 실습 문제에 대한 알고리즘을 기반으로 직접 소스코드를 작성하고 결과 값을 도출해본다. 이러한 블라인드 프로그래밍 실습은 매주 단원별 학습내용에 대하여 1단계→2단계→3단계→4단계 과정을 반복한다.

Fig. 2는 제어문에 해당하는 1단계 응용문제로 '사용자로부터 하나의 정수를 입력 받아 양수, 0, 음수를 판단하는 프로그램'에 대한 문제를 보여주고 있다. Fig. 2의 (a)는 완성된 소스코드를 나타내며, Fig. 2의 (b)는 ①, ②, ③ 부분이 은닉된 상태로 학습자에게 문제로 주어진다. 학습자는 ①, ②, ③에 해당하는 소스코드를 완성하여 실행결과를 테스트해 본다. 이 과정에서 학습자는 해당부분의 코드를 완성하기 위해 자연스럽게 앞에서 배운 내용을 검토하는 작업을 되풀이 하게 된다. 또한, 정상적인 실행결과를 얻기 위해 프로그램을 수정→실행 하는 과정을 반복함으로써 에디터에서 소스코드를 작성하고 인터프리터로 실행하는 작업에 익숙해진다.

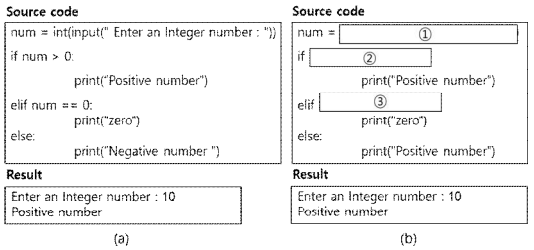

Fig. 2. Example of applied question in Step 1

3.3 기말 프로젝트 수행 방법

학기말 프로젝트 수행단계에서는 프로젝트 계획→알고리즘 작성→프로그램개발 순서로 진행한다. 프로젝트 계획 단계에서는 한 학기동안 학습한 내용을 기반으로 다루어본 예제를 참고하거나 전공분야와 접목시킬 수 있는 주제를 선정하고 전체 프로그램의 콘텐츠를 기획한다. 알고리즘 작성 단계에서는 계획한 내용에 대한 프로그램 동작 과정을 알고리즘화하여 구체적으로 체계화시킨다. 프로그램 개발 단계에서는 작성된 알고리즘을 기반으로 프로그래밍 작업을 하고 다양한 데이터를 입력하여 테스트 작업을 거친 후 프로젝트 작업을 마무리한다[13].

4. 연구방법

4.1 연구대상

본 연구에서 제안한 학습모델은 프로그래밍 관련 교과목을 처음 접하는 학생들을 대상으로 개발되었기 때문에 다양한 학문영역의 학생들을 연구대상으로 편성할 필요가 있다. 따라서 본 연구에서는 경기도 K대학교의 2018년 1학기 교양필수 과목인 '소프트웨어기초' 교과목을 처음 수강하는 비전공 신입생 152명을 연구대상으로 선정하였고 수강생의 전공 및 분포는 Table 2와 같다.

Table 2. Student's Major and distribution of the first semester in 2018

Major	Student (Number)	Percentage (%)
Economics	35	23.0
Intellectual Properties	3	2.0
Accounting·Management Information Systems	19	12.5
Human service	21	13.8
Business Administration	18	11.8
International Trade	18	11.8
Public administration	25	16.4
Law	11	7.2
Police Administration	2	1.3
Total	152	100

강의가 시작된 1주차에 수강생들의 프로그래밍 경험 여부에 대하여 조사하였으며, 그 결과는 Table 3과 같다. 프로그래밍 경험이 있는지에 대한 질문에 92.1%의 학생들이 프로그래밍 경험 없다고 하였다. 그리고 5.3%의 학생들은 프로그래밍을 중·고등학교 때 배웠고, 개인적으로 공부한 경험이 있는 학생들은 2.6%였다. 또한 프로그래밍을 처음 접하는 느낌에 대한 질문에는 "두렵다", "어려울 것 같다", "흥미롭다" 등의 의견을 나타냈다.

Table 3. A survey result of programing experience

Contents	Student (Number)	Percentage (%)
No experience	140	92.1
Learned experience in middle and high school	8	5.3
Learning experience by private study	4	2.6
Total	152	100

4.2 연구절차

본 연구는 DMM모델의 교수학습 절차와 단계적 블라인드 프로그래밍 실습과정을 적용하여 수업을 진행 한 후 주어진 응용문제에 대한 해결 능력의 변화와 학기말에 수행한 프로젝트 내용과 소감문을 통해 프로그래밍에 대한 인식 및 반응을 분석하였다.

수업에서 다룬 프로그래밍 언어는 높은 가독성과 간결한 코딩이 가능한 파이썬(Python) 언어이다[14]. 파이썬은 미국 대학교 등에서도 컴퓨터 프로그래밍 입문 수업으로 많이 사용되는 등 프로그래밍 기법의 원리를 이해하는 코딩 교육에 많이 쓰이고 있다[15]. Table 4는 1주~13주 동안의 학습범위 및 내용을 요약한 것이다.

Table 4. Scope and curriculum of Python programming lecture

Learning Scope	Learning contents
Python Basics	Variables, Input/output function
Operator I, Data	Operators(arithmetic/remainder/square root), Date Types
Lists, Operator II	Lists, Operators(relational/logical)
Control Structures	if/if-else, Conditional expression, Block
Repetition Structures	While/For, Break
Functions	Function definitions and call, Return

1주~13주 동안의 강의진행 방식은 제안한 학습모델의 '이론학습 및 예제연습 → 응용문제 실습' 절차에 따라 학습범위 내의 내용으로 진행하였다. 기말 프로젝트는 14주차에 수업시간동안 배운 파이썬 내용 및 참고자료 범위에서 전공과 연관성 있는 주제 또는 자유주제를 학생-교수자, 학생-학생 간에 토론과정을 통해 선정하였다. 15주차에는 구현 및 테스트 작업을 통해 개발프로그램을 완성하고 프로그램 설명, 알고리즘 및 프로젝트 수행소감을 작성하게 하였다.

5. 연구결과

5.1 응용문제 실습과정 및 결과분석

단계적으로 블라인드 처리된 응용문제는 교과목 교재[16]에서 해당 주차에 대한 내용을 발췌하여 사용하였다. 1주~13주차까지 실습 진행 결과 중 앞 주차에 진행된 내용을 누적하여 단원별 내용을 종합적으로 다룬 4주차(산술, 나머지, 지수연산자, 자료유형), 7주차(관계/논리연산

자, 제어문), 10주차(제어문 응용, 반복문)를 선택하여 분석하였고 결과는 다음과 같다.

각 주차에는 1단계→2단계→3단계→4단계의 응용문제를 각 1문제씩 순차적으로 작성하게 하였다. 응용문제 실습시간에는 단계별 문제별로 아래와 같은 절차에 따라 진행하였으며, 교수자-학습자간 피드백의 유·무에 대하여 기록하였다. 첫째, 학습자 스스로 문제를 해결할 수 있는 시간을 주었다. 둘째, 학습자 스스로 문제해결을 못하는 경우 손을 들어 의사표시를 하게하고 대상자들을 1:1 피드백을 통해 순차적으로 지도하였다. 이때, 단계별 응용문제에 대한 정답은 정확히 해당문제를 작성한 경우 실행결과를 통해 확인할 수 있고, 교재 및 정답을 제시한 강의 자료는 응용문제 실습시간이 지난 후 확인하도록 제한하였다.

Table 5는 총 152명의 학생들을 대상으로 실시한 4, 7, 10주차 수업에서 교수자 피드백 유·무에 따른 단계별 응용문제 해결의 결과이고 각 항목은 명(%)로 나타내었다.

Table 5. Results of applied problem solving with(w/)/without(w/o) professor's feedback, Num of Person(%)

weeks	Feedback	Step				Total
		1	2	3	4	
4	w/o	88 (57.9)	86 (56.6)	80 (52.6)	79 (52.0)	333 (54.8)
	w/	64 (42.1)	66 (43.4)	72 (47.4)	73 (48.0)	275 (45.2)
	Total	152 (100)	152 (100)	152 (100)	152 (100)	608 (100)
7	w/o	105 (69.1)	98 (64.5)	95 (62.5)	86 (56.6)	384 (63.2)
	w/	47 (30.9)	54 (35.5)	57 (37.5)	66 (43.4)	224 (36.8)
	Total	152 (100)	152 (100)	152 (100)	152 (100)	608 (100)
10	w/o	110 (72.4)	103 (67.8)	100 (65.8)	90 (59.2)	403 (66.3)
	w/	42 (27.6)	49 (32.2)	52 (34.2)	62 (40.8)	205 (33.7)
	Total	152 (100)	152 (100)	152 (100)	152 (100)	608 (100)

4주차 수업에서 여러 연산자의 사용방법과 자료유형에 대한 내용을 학습한 후 전체학습자(152명)의 응용문제 실습결과를 살펴본 결과, 1단계 문제는 57.9%, 2단계 문제는 56.6%의 비율로 학생들이 교수자의 피드백 없이 스스로 문제를 해결하였다. 전체 소스코드나 결과 값을 블라인드 처리한 3, 4단계 문제의 경우 52.6%, 52.0%로 1, 2단계 문제에 비하여 학습자 스스로 해결한 비율이 낮게 나타났다.

관계/논리 연산자와 제어문을 학습한 7주차에는 단계별 응용문제에 대해 학습자 스스로 해결한 비율(63.2%)이 4주차(54.8%)보다 평균 8.4% 높게 나타났다. 이러한 현상은 앞 주차에 학습한 연산자 기능을 익힌 상태였으므로 if-else문과 같은 조건식의 사용이 비교적 수월했던 것으로 보인다. 단, 주어진 알고리즘을 기반으로 소스코드를 작성하는 4단계 문제의 경우 56.6%로 7주차의 다른 단계문제에 비하여 낮은 비율을 보였으며, 4주차와 비교했을 때에도 4.6%의 낮은 상승폭을 보였다.

10주차 수업의 단계별 응용문제에 대한 결과도 교수자 피드백 없이 해결한 문제의 비율이 1~4단계 모두 상승한 결과를 보였다. 그러나 7주차→10주차의 상승비율이 평균 3.1%로 4주차→7주차 수업의 평균 상승 비율인 8.4%에 비해 낮은 결과를 나타내었는데, 그 이유를 살펴보면 10주차 수업내용에서 다룬 반복문의 원리와 제어문과 반복문을 혼용한 응용문제에 대해 학습자들이 어렵다는 반응을 보였기 때문이며, 이는 학습자 스스로 해결한 응용문제의 비율이 주차별 학습내용의 난이도에 영향을 받는 것으로 파악된다. 또한, 59.2% 비율을 보인 4단계 문제의 결과를 봤을 때, 학기 후반에 접어든 시점인 10주차에도 여전히 알고리즘을 기반으로 소스코드를 작성하는 방법에 어려움을 느끼고 있는 것으로 파악된다.

Table 5의 열에 나타낸 총합 항목은 주차별 교수자 피드백의 유·무에 따른 학생비율의 변화를 알아보기 위해, 1~4단계 학생 수의 총합(608명)에 대한 교수자 피드백 유·무에 해당하는 각 학생 수의 비율을 계산한 결과이다. 각 주차별 총합 항목을 그래프로 나타내면 Fig. 3과 같다. 4주차에 시행한 응용문제 실습결과를 보면, 각 단계별 문제에 대해 교수자의 피드백을 받지 않고 해결한 비율이 54.8%로 피드백을 받은 45.2% 보다 높게 나타났다. 또한, 7주차 수업에서는 4주차에 비해 평균 8.4% 상승한 비율인 63.2%의 학생들이 스스로 단계별 문제를 해결하는 결과를 보였다. 10주차에는 66.3% 학생들이 교수자의 피드백을 받지 않고 스스로 응용문제를 해결하여 평균 3.1%의 상승률을 나타냈다. 따라서 주차가 진행될수록 학습자 스스로 문제를 해결하는 비율이 상승되고 있음을 확인할 수 있었다.

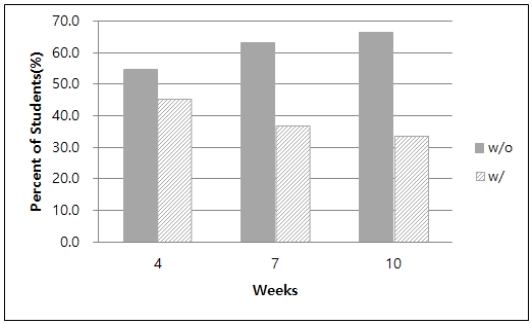

Fig. 3. Distribution of applied problem solving with/without professor's feedback

이러한 단계별 블라인드 처리된 응용문제를 적용한 실습과정은 프로그래밍 수업에서 이론학습과 예제연습에서 끝나지 않고 학생들이 해당주차 내용에 대해 되돌아보며 복습할 수 있게 한다. 또한, 블라인드 처리된 부분을 해결하기 위해 여러 번 코딩을 시도하는 과정에서 자연스럽게 프로그래밍 연습을 할 수 있게 하여 프로그래밍 과목에 친숙해질 수 있는 기회를 제공 할 수 있다.

그러나 반복문과 같이 이해하기 어려운 부분에 대해서는 학습자에 따라 스스로 문제를 해결하는데 한계가 있으므로 교수자의 유연성 있는 피드백이 필요하다. 이때, 교수자는 학생 스스로 해결할 수 있는 능력을 키울 수 있도록 이전에 배운 내용을 연상할 수 있게 역질문을 하고 학습한 내용을 기억하고 응용할 수 있게 유도해야 할 것이다.

5.2 기말프로젝트 수행내용 및 소감문의 정성적 분석

프로젝트 수행내용 및 소감문 분석과정은 전문가 3명(소프트웨어 프로그래밍 분야를 강의하는 교수 2명, 컴퓨터공학 박사 1명)의 검토와 회의를 통해 진행하였다. 프로젝트 수행내용은 학생들의 관심분야, 전공과 관련된 주제 선정, 강의자료 등 참고자료를 기반으로 확장한 주제로 나누어 분류하였고 결과는 Table 6과 같다. 본인의 관심분야 적용 57.3%, 기존에 구현되어 있는 내용을 참고하여 확장 39.5%로 전공과 연관성 있는 경우인 3.3%보다 높은 분포를 보였다. 이는 학생들이 신입생인 관계로 전공에 응용할 수 있는 기초 지식이 부족한 상황이기 때문인 것으로 파악된다.

Table 6. Subjects of performed projects

Subjects	Student (Number)	Percentage (%)
Association with their majors	5	3.3
Application to their interests	87	57.2
Expansion based upon reference materials	60	39.5
Total	152	100

전공과 연관성 있는 주제의 예로는 법학과의 '혈중 알코올 농도 측정 프로그램', 경제학부의 '최대공약수와 최소공배수를 구하는 프로그램' 등이 있었고 관심분야를 적용한 경우는 '메모리 기능 계산기', '학교식당의 포스시스템 구현' 등의 내용으로 구성되어 있었다. Fig. 4는 실제 프로젝트 수행 결과의 사례로 그림4의 (a)는 알고리즘, Fig. 4의 (b)는 실행결과를 나타낸다.

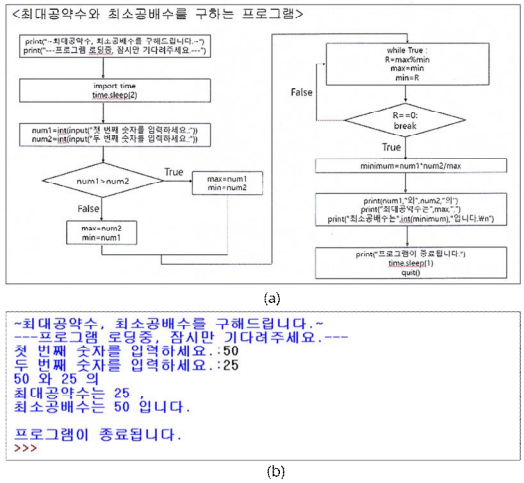

Fig. 4. Example of the executed project result

기말 프로젝트 수행에 대한 소감은 주관식으로 조사하였고, 프로젝트 수행 후 느낀 점에 대한 긍정적인 의견과 부정적인 의견에 대한 사례들을 정리하면 Table 7과 같다.

학습자가 프로젝트 후에 느꼈던 긍정적인 의견은 처음 접해보는 프로그래밍에 대한 흥미와 자신감 및 성취감을 느꼈고 향후 다양한 응용프로그램을 만들어 보고 싶다고 하였다. 이러한 의견을 종합하면, 한 학기 동안 학습한 내용을 기반으로 주제선정에서부터 알고리즘 작성 및 코딩 작업 등 다양한 경험을 하면서 프로그래밍의 활

Table 7. Examples of what you felt after you performed the project

Opinion	Contents
Positive	I have an interest in programming and want to continue studying Python programming techniques.
	After completing the project, I feel that my understanding of the program is increased and become confident.
	It is amazing to be able to program real life problems after project, and I want to make various applications in the future.
Negative	There was a function that I wanted to create, but I was sorry that I couldn't implement everything with what I learned in class.
	The programming itself was so difficult that the project work was hard and burdensome.
	It was difficult to complete the project because there was not enough time to do the actual programming work by modifying the project theme in the middle.

용방법을 익히고 계속적으로 공부해보고 싶은 동기가 부여된 것으로 사료된다.

프로젝트 수행이 어렵게 느껴지는 등 부정적인 측면에서는 한 학기(3학점 이내) 동안 진행되는 수업내용으로 프로그래밍의 다양한 기능을 모두 다루어 볼 수 없기 때문에 개발범위의 한계를 느끼고 있었다. 반면, 학기말이 되어서도 프로그래밍 작업을 힘들어하며 프로젝트 수행에 부담을 느끼는 것으로 파악되었다. 또한, 정해진 시간동안 진행되는 프로젝트 작업 과정에서 주제선정을 변경하는 등 문제가 발생되는 경우 시간이 부족하여 어려움을 느끼는 것으로 나타났다. 이를 통해 수업난이도의 조절, 학습자간 이해도의 편차를 조율하기 위한 방안이 모색되어야 할 것으로 보인다.

6. 결론

본 연구에서는 소프트웨어 기초교육을 위한 학습모델을 설계하고 프로그래밍을 처음 접하는 비전공 신입생들을 대상으로 한 실제 수업에 적용한 후 학습자의 문제해결 능력의 변화와 프로그래밍에 대한 인식 및 반응에 대해 심도 있게 분석하였다. 제안한 학습모델을 기반으로 수업을 진행하며 단계적 블라인드 처리된 응용문제에 대해 교수자 피드백의 유·무에 따른 학습자의 문제해결 능력을 관찰한 결과, 교수자의 피드백 없이 문제를 스스로 해결한 학생의 비율이 주차가 지날수록 54.8%(4주차), 63.2%(7주차), 그리고 66.3%(10주차)로 점차 증가하는 결과를 보였다. 즉, 주차가 진행될수록 학습자 스스로 문제해결을 하는 비율이 상승되는 결과를 확인하였다. 또한, 기말 프로젝트 수행내용과 소감문의 사례들을 토대로 프로그래밍 작업에 대한 긍정적인 의견과 부정적인 의견을 분석하여 컴퓨터 및 프로그래밍 작업에 익숙하지 않은 비전공 신입생을 대상으로 하는 교양 수업을 계획함에 있어 수업난이도의 조절, 학습자간 이해도의 편차를 조율하기 위한 방안이 모색되어야 한다는 의미 있는 결과를 도출하였다. 또한 이 연구 결과를 통해 비전공자들이라 하더라도 소프트웨어 기초학습에 흥미를 유발하고 동기를 부여할 수 있음을 알 수 있었다. 그러나 제안한 방법은 학습자들이 가진 이해력과 실습 속도에 차이가 있기 때문에 제한된 시간 내에 교수자의 피드백을 필요로 하는 모든 학습자들을 충분히 지도하는데 시간적 한계가 존재한다. 이러한 결론을 바탕으로 향후 연구에서는 보다 다양한 측면에서 학습자들의 학습성과를 분석하고, 이를 기반으로 학습내용의 난이도와 학습자 이해도의 편차를 고려한 효율적인 교수법에 대한 연구가 필요할 것이다.

REFERENCES

[1] B. E. Penpras. (2018). The Fourth Industrial Revolution and Higher Education. Palgrave Macmillan. https://link.springer.com/chapter/10.1007/978-981-13-0194-0_9

[2] Harvard University. https://cs50.harvard.edu

[3] Columbia University. https://www.cs.columbia.edu/education/courses

[4] J. Kim. (2017, September). Software Education for Non-Majors and Sungkyun SW Education iNstitution operation case. *The 10th joint deliberation for professors and faculty members in charge of liberal education.*. (pp. 29-49). Seoul : Korea National Institute for General Education.

[5] J. Seo. (2017). A Case Study on Programming Learning of Non-SW Majors for SW Convergence Education. *Journal of Digital Convergence, 15(7),* 123-132. DOI : 10.14400/JDC.2017.15.7.123

[6] H. Kim. (2015, November). Trends of Computational Thinking Education in University. *KERIS Symposium in ICT education.* (pp. 237-250). Daegu : Korea Education & Research Information Service.

[7] National IT Industry Promotion Agency.

http://www.software.kr/um/um03/um0303/um030302/um03030203.do

[8] S. Y. Pi. (2016). A Study on Coding Education of Non-Computer Majors for IT Convergence Education. *Journal of Digital Convergence, 14(10),* 1-8. DOI : 10.14400/JDC.2016.14.10.1

[9] S. H. Park. (2016). Study of SW Education in University to enhance Computational Thinking. *Journal of Digital Convergence, 14(4),* 1-10. DOI : 10.14400/JDC.2016.14.4.1

[10] S. Lee & J. Kim. (2017, August). A Study of SW Education for Non-Majors in Sungkyun SW Education iNstitution(SSEN). *Proceedings of the Korea Association of Computer Education.* (pp. 107-109). KACE.

[11] Y. Lee. (2018). Python-based Software Education Model for Non-Computer Majors. *Journal of the Korea Convergence Society, 9(3),* 73-78. DOI : 10.15207/JKCS.2018.9.3.073

[12] J. Kim et al. (2015). *A Research on the Development of Teaching and Learning Models for SW Education.* Seoul : Korean Educational Development Institute.

[13] H. W. Jung. (2019). A Study on Teacher-learner Feedback Method for Effective Software Project Execution of Non-Computer Major Students. *The Journal of the Convergence on Culture Technology(JCCT), 5(1),* 211-217. DOI : 10.17703/JCCT.2019.5.1.211

[14] Python Software Foundation. https://www.python.org/about

[15] IEEE Spectrum. https://spectrum.ieee.org/computing/software/the-2018-top-programming-languages

[16] I. G. Chun. (2017). *pitapat python.* Paju : Life & Power Press.

정 혜 욱(Jung, Hye Wuk) [정회원]

- 1999년 2월 : 한성대학교 정보전산학부 (공학사)
- 2005년 2월 : 성균관대학교 정보보호학과 (공학석사)
- 2013년 2월 : 성균관대학교 컴퓨터공학과 (공학박사)
- 2015년 11월 ~ 2018년 2월 : 성균관대학교 컨버전스연구소 선임연구원
- 2018년 3월 ~ 현재 : 경기대학교 융합교양대학 교양학부 조교수
- 관심분야 : 소프트웨어 교육, 지문분류, 인공지능, 딥러닝, 이미지검색, 정보보호
- E-Mail : wukj@kyonggi.ac.kr