JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 30 No. 11, pp. 417-429, November 2025
https://doi.org/10.9708/jksci.2025.30.11.417

Systematic Design and Validation of Programming Education Prompts Utilizing LLMs to Enhance College Students' Computational Thinking Skills

Seul-ki Kim*

*Teacher, Elementary School, Gyeonggi-do Office of Education, Gyeonggi-do, Korea

[Abstract]

This study aimed to develop a sustainable programming education solution utilizing LLMs by designing a Programming Education Prompt (PEP) based on pedagogical perspectives and validating its effectiveness. While existing LLM-based studies have shown limitations in cost and scalability issues by focusing on the lack of pedagogical perspectives and the development of individual API-based services, this study presented a solution that can be utilized without restrictions in all educational environments through a prompt-centric approach. Applying Design-Based Research methodology, the study went through iterative cycles of design, implementation, analysis, and redesign, with a group of 12 experts participating in the development and validation of the prompts. The developed PEP consisted of three domains: concept learning, programming exercise, and debugging, and expert review confirmed high validity in all domains (CVR=0.83-1.00). In a 15-week quasi-experiment (N=46), the PEP group showed statistically significant improvement in algorithmic thinking compared to the control group (p=.010). In the evaluation domain, statistical significance was not achieved after multiple comparison correction (p=.022), but a medium-sized practical effect was identified (δ=0.41). This suggests that LLM-based programming education should be implemented through systematic design from a pedagogical perspective beyond simple utilization, and demonst.

▸ **Key words**: Programming Education, LLM, Prompt, Informatics Education, Computational Thinking

[요 약]

본 연구는 LLM을 활용한 지속 가능한 프로그래밍 교육 솔루션 개발을 목적으로, 교육적 관점에 기반한 프로그래밍 교육용 프롬프트(Programming Education Prompt, PEP)를 설계하고 그 효과성을 검증하였다. 기존 LLM 활용 연구들이 교육적 관점 부족과 API 기반 개별 서비스 개발에 집중하여 비용 및 확산 문제 등의 한계를 보인 반면, 본 연구는 프롬프트 중심 접근을 통해 모든 교육 환경에서 제한 없이 활용 가능한 솔루션을 제시하였다. 설계 기반 연구(Design-Based Research) 방법론을 적용하여 설계, 실행, 분석, 재설계의 반복적 순환 과정을 거쳤으며, 12명의 전문가 집단이 프롬프트 개발과 검증에 참여하였다. 개발된 PEP는 개념 학습, 프로그래밍 연습, 디버깅의 세 영역으로 구성되었으며, 전문가 검토 결과 모든 영역에서 높은 타당도를 확보하였다(CVR=0.83-1.00). 15주 준실험(N=46)에서 PEP 집단은 알고리즘적 사고 영역에서 통제집단 대비 통계적으로 유의한 향상을 보였다(p=.010). 평가 영역에서는 다중비교 보정 후에는 통계적 유의성에 미치지 못하였으나(p=.022) 중간 크기의 실질적 효과를 확인할 수 있었다(δ=0.41). 이는 LLM 기반 프로그래밍 교육이 LLM의 단순 활용을 넘어 교육적 관점의 체계적 설계를 통해 구현되어야 함을 시사하며, 프롬프트 중심 접근이 다양한 교육 환경에서 지속 가능한 대안임을 보여준다.

▸ **주제어**: 프로그래밍 교육, LLM, 프롬프트, 정보 교육, 컴퓨팅 사고력

• First Author: Seul-ki Kim, Corresponding Author: Seul-ki Kim
*Seul-ki Kim (tmfrlska85@gmail.com), Elementary School, Gyeonggi-do Office of Education
• Received: 2025. 09. 23, Revised: 2025. 10. 20, Accepted: 2025. 11. 03.

Copyright © 2025 The Korea Society of Computer and Information
http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945
www.kci.go.kr

I. Introduction

디지털 사회로의 전환이 가속화됨에 따라 컴퓨팅 사고력의 중요성이 높아지고 있다. 컴퓨팅 사고력의 정의는 학자들 마다 다양하게 나타나지만 컴퓨터 과학의 개념을 사용하여 문제를 해결하는 능력이라는 공통된 의미를 포함하고 있다[1-3].

최근 미래 사회를 살아갈 학생들의 기초 역량을 길러주기 위한 관점에서 컴퓨팅 사고력을 신장 시킬 수 있는 방법에 대한 연구가 꾸준히 이루어졌다. 관련된 여러 선행 연구에서 컴퓨터를 사용하여 문제를 해결하고 정보를 처리하는 능력을 갖추는 것이 필수적임을 강조하고 있으며 이를 위한 컴퓨터 기초 교육의 필요성을 주장하고 있다[5]. 특히 프로그래밍 교육이 컴퓨팅 사고력을 길러 줄 수 있는 효과적인 교육 방법 중 하나로 강조되고 있으며 학습자의 문제 해결력과 논리적 사고력을 배양할 수 있는 중요한 교육 방법으로 제안되고 있다[6,7].

이러한 중요성과 장점에도 불구하고 프로그래밍은 많은 기술과 다양한 지식을 포함하는 복잡한 작업이기 때문에 기초 학습자에게 어려움을 주는 것으로 알려져 있다[8-10]. 이러한 교육 현장의 어려움을 극복하고자 다수의 선행 연구에서 학습자의 인지적, 정의적 차이로 부터 유발되는 요인들을 도출하고 이를 해결하기 위한 방안을 제시하고 있다. 하지만 여전히 교육 현장에서는 대규모 교육 환경에 적용하는 데 어려움을 겪고 있으며, 개념적 지식에 대한 의존성 및 교육 프로그램의 전반적인 개편의 어려움으로 인해 실제적으로 적용되지 못하는 한계가 지적되고 있다[11,12].

최근 교육 현장에서 프로그래밍 교육을 적용할 때 발생하는 다양한 유형의 어려움을 해결하고 지속 가능한 프로그래밍 교육에 필수적인 교수학습 환경을 구축하기 위해 LLM(Large Language Model)을 활용하는 접근이 다양한 관점에서 연구되고 있다. 주로 LLM을 활용하여 코드에 대한 설명을 생성하고, 학생이 생성한 결과와 비교하여 그 유용성을 확인하였으며, 초보자의 기초 프로그래밍 학습을 지원하기 위해 다양한 수준의 힌트를 제공하는 인터페이스를 개발하여 적용한 연구 사례를 살펴볼 수 있다[13,14].

이러한 선행 연구들은 대부분 LLM 모델의 API를 사용하여 개별 서비스 혹은 인터페이스를 구성하고 그 적용 가능성을 확인하였다. 따라서 대규모 적용에 대한 유지 보수 측면에서의 비용 문제가 발생할 수 있으며 일반적인 교육 환경에 적용할 수 없고 지속 가능한 프로그래밍 교육을 지원할 수 없다는 한계가 있다. 또한 프로그래밍 교육 방법과 교육학적 접근이 반영되지 않아 교육의 측면에서의 논의가 부족하다는 한계점 또한 확인할 수 있다. 이에 본 연구는 프로그래밍 교육 및 LLM의 효과적인 활용과 관련된 선행 연구를 분석하고 교육적 관점을 기반으로 LLM을 활용하는 방안을 연구하고자 한다. 이를 통해 광범위한 교육 현장에 적용할 수 있으며 학습자의 컴퓨팅 사고력을 효과적으로 길러줄 수 있는 지속 가능한 프로그래밍 교육 솔루션을 제공하는데 목적이 있다.

II. Preliminaries

1. Computational Thinking and Programming

컴퓨팅과 컴퓨터 과학 교육의 중요성은 컴퓨팅 사고력이라는 명칭의 정의가 논의되기 전부터 다방면으로 연구되어 왔다. Denning(2007)은 컴퓨팅을 자연 과학의 한 부분으로 설명하며 계산, 통신, 조정, 회상, 자동화, 평가, 설계로 구성된 프레임워크를 제안하였다. 그리고 컴퓨팅은 새로운 원칙과 기술의 지속적인 발견으로 진화하며 기본적인 컴퓨팅 지식을 가르칠 수 있는 교육의 대중화가 필요함을 강조하였다[1]. 이후 Wing(2008)의 연구로 컴퓨팅 사고력(Computational Thinking)이라는 명칭이 본격적으로 사용되면서 다양한 관점에서의 정의와 관련 연구들이 뒤이어 이루어졌다. Wing은 컴퓨팅 사고력을 복잡한 문제를 공식화하고 불필요한 세부 사항을 제거하여 더 간단하게 해결할 수 있는 문제로 변환하며 이를 자동화하여 처리할 수 있는 능력으로 정의하였다. 이는 문제 해결 접근 방식과 인간이 문제를 해결하는 방식을 이해하는 데 중요한 역할을 한다고 설명하며 추상적이고 고차원적인 사고의 중요성을 함께 강조하였다[2].

프로그래밍과 사고력을 중심으로 심도있게 분석한 선행 연구에서는 프로그래밍 언어는 인간의 언어나 수학의 공식과는 다르며, 이론과 실습이 밀접하게 통합된 과정으로 설명되고 있다. 특히 프로그래밍은 엄격하고 구조적인 특징을 가지고 있어, 컴퓨팅 사고력의 구조적이며 논리적인 사고를 반영하기 적절한 교육 방법임을 강조하고 있다. 이러한 특징으로 학생들은 프로그래밍을 통해 다양한 문제 해결 방법을 배우고 다각적으로 사고를 확장시킬 수 있다는 이점을 제시하였다[10,11].

컴퓨팅 사고력의 정의는 학자들 마다 다양하게 나타나지만 대체로 컴퓨터 과학의 기본 개념을 활용하여 문제를 해결하고 시스템을 설계하며, 인간의 행동을 이해하는 능력을 포함하는 공통적인 관점을 공유하고 있다. 특히 컴퓨

www.kci.go.kr

팅 시스템을 설계하거나 활용하기 위한 기본 도구로서 프로그래밍이 중요한 도구 및 교육 방법으로 제시되고 있음을 확인 할 수 있다.

2. Challenges of Programming Education

많은 연구자들이 프로그래밍 교육의 장점을 논의하고 교육 현장에 도입하기 위한 방안을 연구했지만, 여전히 교사들은 교육 현장에서 발생하는 다양한 이유로 인해 프로그래밍 교육 적용에 어려움을 겪고 있다. 먼저 학습자 내부적 측면에서의 어려움과 관련된 연구에서는 주로 프로그래밍 학습 시 나타나는 단순한 문법적 오류와 이를 해결하는 과정에서 학습자들이 느끼는 부담감, 추상적인 개념과 알고리즘의 복잡성이 프로그래밍 교육의 어려움을 유발하는 주요 원인으로 제시되었다[9,15]

학습자 외부적 측면의 어려움에 대한 연구에서는 비실용적이며 학생의 관심을 끌지 못하는 수업 형태의 문제가 지적되었으며 교육 대상자의 규모가 커지면서 교육자와 학습자의 상호작용이 활발하게 이루어지지 못하는 문제 및 교수학습 과정 중 개별 학습자에게 주어지는 불평등한 기회 제공 등이 문제점으로 제기되었다[11,16].

이러한 문제점을 해결하기 위해 프로그래밍 교육을 위한 다양한 방법론이 연구되었다. 선행 연구에서는 주로 학습자 중심의 교수학습 모델을 통해 과제에 대한 적극적 참여와 긍정적 성공 기대를 높이는 방향이 중요함을 강조하고 있다[17].

교수학습 환경 연구에서는 삶의 맥락과 관련된 주제 선정과 흥미로운 사례를 통한 참여 촉진이 필요하며, 실습 중심 환경 구성이 필수적으로 강조된다. 또한 지나치게 간결하고 추상적인 코드보다는 가독성이 높고 교육을 목적으로 설계된 예시 코드가 중요함을 확인할 수 있다[18].

프로그래밍 교육 순서에 관한 연구에서는 코드 추적, 코드 조작, 코드 작성 순으로 프로그래밍 능력이 발달함을 확인하였다. 이를 위한 교육 방법으로 프로그래밍 템플릿 제공, 단계적 소스 코드 활용, 파슨스 문제(Parsons Problems) 등이 기초 학습자에게 적합함을 제시하였다[19]. 특히 파슨스 문제와 선택적 인터페이스를 통해 기존 방법의 한계를 개선한 연구에서는 CS1 교육과정 적용 결과 학생들의 프로그래밍 패턴 습득과 기술 향상에 긍정적 영향을 확인하였다[8].

프로그래밍 교육을 위한 다양한 교육 방법이 실제 교육 현장에 정착되기 위해서는 교육 방법이 적용된 프로그래밍 예제 및 문제 개발이 필수적으로 선행되어야 한다. 하지만 교수자 개인이 수업과 관련된 모든 리소스를 생성하는 데에는 한계가 있으며, 학생 등 외부에서 생성된 리소스 또한 품질과 관련된 문제가 발생할 수 있다. 따라서 양질의 리소스 개발을 위한 방법적인 측면에서의 연구가 함께 이루어질 필요가 있다[9].

3. LLM for Programming Education

방대한 언어 데이터를 학습한 LLM은 인간의 언어를 확률적으로 이해하고 최적의 응답을 생성하여 다양한 산업 분야에서 활용되고 있다[21,22]. LLM의 텍스트 생성 능력은 코드 생성으로 확장되어 자연어를 통해 코드를 생성하거나 코드를 통해 설명을 생성하는 등의 프로그래밍을 지원하는 도구로 활용되고 있다[23]. 선행 연구에서는 LLM이 복잡한 개념 이해, 코드 작성 문제 해결을 위한 설명 및 예제 생성, 오류 수정과 코드 최적화에 활용 가능함을 확인하였다[24]. 또한 효과적인 코드 생성을 위해서는 명확하고 간결한 질문, 과제의 맥락과 세부 정보 제공이 포함된 프롬프트 설계가 중요성을 강조하였다[25,26].

프로그래밍 교육에서의 LLM 활용 연구도 주로 API를 통한 코드 설명, 오류 수정, 힌트 제공 방향으로 이루어지고 있으며 프로그래밍 역량과 컴퓨팅 사고력 향상에 긍정적인 영향을 미치고 있음을 확인할 수 있다[14,27].

이러한 긍정적인 측면과 함께 LLM 활용의 부정적인 측면도 함께 연구되고 있다. 선행 연구에서는 LLM이 코드 문제를 일관되게 식별하지 못하고 존재하지 않는 문제를 지적하는 비율이 높으며, 즉각적인 피드백이 깊이 있는 학습을 저해할 수 있다는 문제점을 도출하였다[27,28]. 또한 학생들의 과도한 의존성, 응답 부정확성으로 인한 방해, 프롬프트 형태에 따른 질적 차이, 기초 학습자에게 적합하지 않은 복잡성과 추상성 등의 한계가 확인되었다[13,29]. 추가적으로 API 사용에 따른 비용 문제와 구조화된 인터페이스로 인한 유연성 부족도 대규모 교육 환경에서의 지속적 활용을 제한하는 요소로 제시되었다[28,29].

선행 연구를 종합하면 LLM은 컴퓨팅 사고력을 기를 수 있는 프로그래밍 교육 지원 도구로서 높은 활용성을 가지고 있으며 교육의 목적에 따라 체계적으로 설계된 프롬프트가 필수적으로 연구되어야 함을 도출할 수 있다. 이에 본 연구는 프로그래밍 교육 교수학습 환경에서 발생할 수 있는 다양한 어려움 지원을 목적으로, 교육적 관점에서 효과적인 프로그래밍 교육 방법을 적용하고 대규모 교육 환경에서 제한 없이 활용될 수 있는 프로그래밍 교육용 프롬프트를 개발하고자 한다. 특히 학습자가 주도적으로 학습하고 개별 수준에 적합한 문제를 해결하는 경험을 가질 수 있는 프로그래밍 교육 콘텐츠를 생성하는데 초점을 맞춰

www.kci.go.kr

프롬프트를 개발하고 실제 교육 현장에 적용하여 효과성을 확인하고자 한다.

III. Research Methods

본 연구는 교육적으로 효과적인 응답을 생성할 수 있는 프로그래밍 교육용 프롬프트(Programming Education Prompt, PEP)를 개발하고 그 효과성을 분석하기 위해 설계 기반 연구 방법(Design-Based Research)을 사용한다. 세부 절차로 설계, 실행, 분석 및 재설계의 지속적인 순환 과정을 포함하여 (Fig. 1)과 같이 구성하였다.

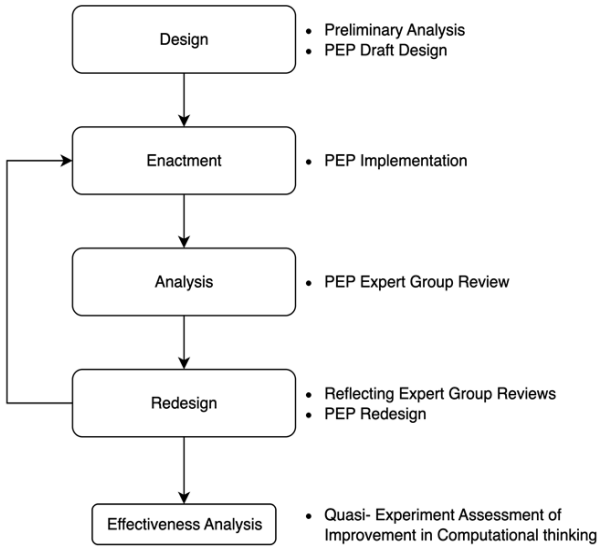

Fig. 1. Research Process

설계 기반 연구는 교육적 개입을 적용하며 복잡하고 실제적인 학습 환경 내에서의 효과를 연구하는데 최적화된 실용적인 접근이다. 또한 이론과 실제 사이의 간극을 메우고, 교육 현장에서 실질적으로 활용 가능한 지식과 정교화된 결과물을 창출하는 데 그 목적이 있다[30,31]. 이에 프로그래밍 교육 현장에서 활용 가능한 LLM을 체계적으로 개발하고 효과성을 분석하는 본 연구의 목표에 가장 부합하는 방법론이라고 할 수 있다.

1. Research Participants

연구 참여자는 연구 단계에 따라 두 집단으로 나눌 수 있다. 첫 번째 집단은 전문가 집단으로 설계 기반의 핵심인 지속적인 순환 과정에 주도적으로 참여한다. 설계 기반 연구는 다학제적 협력을 강조하며 순환 과정에서 학문적 경계를 넘나드는 전문성 교환의 과정을 중시한다[31]. 이에 <Table 1>과 같이 컴퓨터 교육, 컴퓨터 공학, 교육 공학 전공 석사 학위 이상을 가지며 교육 경력 10년 이상의 교사와 대학교수 및 연구자로 구성된 12명의 전문가 집단을 구성하였다.

Table 1. Expert Group Details

Category		Number of Experts
Major	Computer Science	3
	Computer Education	6
	Educational Technology	3
Final Degree	Master	5
	Doctor	7
Education Experience	10 ~ 15 years	4
	15 ~ 20 years	8
Education BackGround	University	7
	High School	5

두 번째 집단은 최종적으로 개발된 PEP를 적용하고 효과성 검증 및 설계 원리와 맥락화 된 이론을 도출하기 위한 준실험 연구에 참여하는 집단이다. K대학교의 컴퓨터 교육 전공 대학생 중 C언어 프로그래밍 수업을 듣는 학생을 대상으로 하였으며 연구 참여자의 프로그래밍에 대한 사전 경험 수준을 통일하기 위해 전체의 학생 중 C언어 관련 선행 교육의 경험이 없는 학생만 샘플링하여 실험집단과 통제집단 각각 23명으로 구성하였다.

2. Research Tools

설계 기반 연구는 연구 참여자들의 긴밀한 협력과 소통을 중심으로 문제를 해결하고 교육적 개입을 개선하기 때문에 객관적인 평가 방법론이 함께 도입되어야 한다[31]. 이에 분석 단계에서 PEP 타당성을 정량적 및 정성적으로 평가하기 위해 설문과 비대면 인터뷰를 진행한다. 설문은 PEP의 주요 요소에 대한 적합도와 생성된 응답의 정확도를 답할 수 있도록 구성하였으며 적합도는 '필수적임', '유용하지만 필수적이지 않음', '불필요함' 3점 척도를 활용하여 기재할 수 있도록 하고 정확도는 5점 척도로 응답할 수 있도록 하였다. 적합도에 대한 응답 결과는 '필수적임'을 동의로 판단하며 전문가의 합의 수준을 정량적으로 비교하기 위한 방법인 CVR(Content Validity Ratio)을 활용하여 평가한다. 또한 Lawshe(1975)가 제시한 최소 기준에 따라 본 연구에 참여한 전문가 수인 12명의 임계치인 0.56 이상을 기준으로 판단한다[32].

PEP의 효과성을 검증하기 위한 도구는 Kılıc 외(2021)에 의해 개발된 Scale for Developing

www.kci.go.kr

Programming-Oriented Computational Thinking 를 활용한다. 이 척도는 프로그래밍에 초점을 맞춘 컴퓨팅 사고력을 측정하기 위한 목적으로 개발되었으며 선행 연구를 통해 높은 신뢰도(Cronbach α=.970)와 내적 타당도를 확보하여 본 연구의 컴퓨팅 사고력을 평가하기에 적합한 도구이다[33]. 이 척도는 개념적 지식과 알고리즘적 사고, 평가의 하위 요인으로 구성되어 있으며 각 하위 요인의 정의와 하위 문항 수는 <Table 2>와 같다.

Table 2. Scale for Developing Programming Oriented Computational Thinking

Factor (Number of Question)	Description
Conceptual knowledge (12)	Conceptual knowledge refers to understanding and using programming structures, including the syntax of programming languages and the process of breaking down problems into meaningful parts and focusing on core functions
Algorithmic thinking (7)	The skill of designing meaningful structures when solving problems using programming concepts, including the ability to understand, apply, evaluate, and create algorithms
Evaluation (14)	Refers to the process of addressing the suitability of algorithms, coding, programming, and systems, including items such as error evaluation, debugging, effectiveness assessment, generalization, and result prediction

IV. Research Results

1. Design and Enactment

PEP 개발을 위한 LLM은 비영어권 언어에서도 상대적으로 높은 성능을 보이며 코드 작성과 관련된 다양한 선행 연구에서 많이 활용된 OpenAI의 모델을 활용한다[27]. 프롬프트 개발을 위해 gpt4-o모델의 API를 활용하여 초안 프롬프트를 입력하고 반복적으로 100개의 응답을 생성하여 전체적인 콘텐츠의 질을 분석한다. 또한 프롬프트를 구성하는 단어를 미세 수정하며 응답을 확인하고 개선하는 작업을 반복하여 <Table 3>과 같이 초안을 개발하였다.

Table 3. PEP Draft

Type	Prompt
Role Setting	You must support C language programming learning from an educational perspective. Understand the following points and respond to the request accordingly: ▪Use a friendly and encouraging tone, like an educator ▪Explain using real-life examples ▪When generating code, prioritize readability above all else.
Concept Learning	Explain C language concepts using real-life examples and provide sample code. Responses should follow the [Template] below. [Template] ### Concept Explanation: Basic concept description ### Real-life Example: Example where the concept is used ### Sample Code: Short, highly readable code ### Notes: Important considerations for the concept or code
Programming Exercise	Generating a Parsons problem where learners arrange code sequences using (concept) in order of difficulty (Level). Respond according to the [Template] below. [Template] ### Problem: [Real-life scenario problem] ### Algorithm Description: [Algorithm explanation] ### Parsons Problem: [Present scrambled code]
Debugging	Provide (Hint Type) to improve the code below, following the [Template] below. ▪Directional Hint: Detailed instructions on the area to focus on ▪Tool-based hint: Briefly describe the task to be performed ▪Example task hint: Provide similar example code (Code) (Improvements) [template] ### Hint Type: [Hint Type] ### Suggestions: [Provide assistance based on the hint type without revealing the correct answer]

선행 연구 분석을 통해 효과적인 코드 생성을 위해서는 명확하고 간결한 요청, 과제의 맥락과 세부 정보 제공이 중요함을 확인할 수 있다[25,26]. 교육 목적의 맥락을 효과적으로 제공하기 위해 페르소나 패턴 원리를 적용하고 교육자라는 직업을 키워드로 설정하였다[34]. 또한 전체 대화에서 학습자의 이해도를 높이기 위해 쉬운 예시와 함께 코드를 제시할 때는 가독성이 높은 코드로 제시할 수 있도록 요구 사항을 명시적으로 구성하였다[18]. 또한 LLM의 확률적 생성 특징 속에서도 정형화된 응답을 얻을 수 있도록 출력 결과를 제한할 수 있는 템플릿 패턴을 활용하여 전반적인 응답 결과의 질을 높이고자 하였다[34].

www.kci.go.kr

개념 학습 프롬프트는 주요 소재로 학생들의 삶의 맥락과 관련 있는 주제의 선정이 필요하며 흥미로운 사례 포함을 통해 참여를 촉진해야 한다는 선행 연구를 근거로 생활 속 사례라는 키워드를 포함하여 구성하였다[5].

프로그래밍 연습 문제에서는 논리적 사고를 촉진하고 문법의 이해를 증진시켜 줄 수 있으며 상대적으로 인지부하를 줄일 수 있는 파슨스 문제 형태로 생성할 수 있도록 프롬프트를 구성하였다[19].

디버깅 프롬프트는 LLM이 학생들의 코드 오류를 수정할 수 있는 방법을 제안할 수 있도록 구성하였다. 특히 LLM이 직접적으로 코드를 수정하지 않도록 하며 선행 연구 분석을 통해 프로그래밍 교육을 위한 힌트 중 방향의 힌트와 도구적 힌트, 작업 예제 힌트 유형을 활용하여 선택적으로 적용할 수 있도록 하였다[14].

2. First Analysis Results

전문가 집단에게 PEP를 직접 시연하고 2주간의 시간 동안 활용 후 적합성을 평가할 수 있는 설문에 대한 응답을 요청하였다. 12명의 전문가가 모두 응답하였으며 응답 결과는 <Table 4>와 같다.

Table 4. Results of First Experts Review

Validity	Agreements	CVR
Role setting	12	1.00
Concept Learning	10	0.67
Programming Exercise	9	0.50
Debugging	11	0.83
Accuracy	M	SD
Concept Learning	4.33	0.70
Programming Exercise	3.25	1.09
Debugging	4.42	0.75

또한 전문가 비대면 인터뷰를 진행하여 8명의 전문가와 실시간으로 의견을 교환하는 인터뷰를 진행하였으며 4명의 전문가는 서면으로 의견을 제시하였다. 전문가 설문과 인터뷰를 통한 시사점은 다음과 같다.

먼저 해당 대화 세션에서 LLM의 전체적인 역할을 부여하는 역할 설정 프롬프트의 적합성이 매우 높게 나타났으며 전문가 인터뷰에서 일부 더 상세한 맥락이 추가될 필요가 있다는 의견이 있었다.

개념 학습 프롬프트의 검토 결과(적합도 CVR=0.67, 정확도 M=4.33) 또한 CVR 값의 결과가 기준 이상(CVR>0.56)으로 나타났다. 세부적으로 생활 속 예시를 중심으로 간단한 코드와 함께 개념을 학습할 수 있는 콘텐츠를 제시하는 형태는 매우 적절했으며 실제 LLM의 응답 또한 매우 정교하게 생성되어 활용도가 높을 것이라는 의견을 확인할 수 있었다. 추가적으로 초보 학습자의 수준을 고려하여 개념 학습 시 유의해야 할 점을 함께 생성한다면 학습에 더 도움이 될 수 있을 것이라는 의견이 있었다.

프로그래밍 연습 적합도와 응답의 정확성은 다른 영역에 비해 비교적 낮은 결과를 보였다(적합도 CVR=0.50, 정확도 M=3.25). 특히 판단 기준으로 제시된 CVR 값(CVR>0.56) 미만의 결과가 나타났다. 전문가 인터뷰 결과 생성된 파슨스 문제의 경우 생성되는 코드 라인의 양식이 세션마다 큰 차이를 보였으며 코드 라인이 제시되지 않는 경우가 많아 학습자가 혼란을 겪을 수 있을 것이라는 의견이 나타났다. 다수의 전문가가 파슨스 문제 형태의 도입은 매우 적절하지만 더 상세한 프롬프트 설정을 통해서 학습자에게 적절한 문제를 제공하는 것이 필요하다는 의견을 제시하였다. 추가적인 의견으로 학습자가 본인의 수준을 입력하는 키워드가 제시되고 있는데 초보 학습자는 수준을 표현하는 키워드를 입력하는데 어려움을 겪을 수 있기 때문에 프롬프트 제시할 때 임의의 수준을 함께 제시해서 선택적으로 활용할 필요가 있다는 의견이 제시되었다.

디버깅 프롬프트의 결과는 CVR 판단 기준 이상(CVR>0.56)으로 나타났으며 가장 높은 적합도와 정확도의 결과를 확인할 수 있었다(적합도 CVR=0.83, 정확도 M=4.42). LLM의 생성 결과가 비교적 정확하게 나타났으며 실제 해결책을 제시하지 않아 교육적으로 효과성이 높을 것이라는 긍정의 평가들이 제시되었다. 추가적인 의견으로 사용자가 코드와 에러 메시지를 함께 입력할 수 있는 구성이 필요하며 단순히 코드의 오류 수정 뿐만 아니라 효율적인 코드 작성을 위한 개선에도 활용될 수 있도록 추가적인 구성이 필요하다는 의견을 확인할 수 있었다.

마지막으로 모든 프롬프트에서 문장의 구성보다는 개조식으로 작성하고 효율적인 프롬프트의 지시 구성이 필요하다는 의견도 제시되었다.

3. Redesign and Enactment

전문가 검토 및 인터뷰 결과를 바탕으로 새롭게 구성된 프롬프트는 <Table 5>와 같다.

www.kci.go.kr

Table 5. Revised PEP

Type	Prompt
Role Setting	You must support C language programming learning from an educational perspective. Understand the following points and respond to the request accordingly: **Basic Background Settings** ▪Use tone appropriate for beginner learners ▪Provide detailed steps that beginners can follow ▪Prioritize explaining fundamental principles of concepts ▪Utilize real-life examples **Prohibited Items** ▪Prohibit use of C++ or other language syntax ▪Prohibit optimization algorithms beyond beginner comprehension
Concept Learning	Explain C language (concepts) using real-life examples and provide sample code. Responses should follow the [Template] below. [Template] ### Concept Explanation ▪Core Definition: Simple definition ▪Operating Principle: How it works ▪Purpose of Use: Explanation with real-life cases ### Sample Code ```c ▪Minimal code showing only core concepts ▪Variable names: Clear English with meaningful names ▪Comments explaining key lines ``` ### Beginner Precautions ▪Most common error types beginners encounter ▪Points to check when errors occur
Programming Exercise	Generating a Parsons problem using (concept) in order of difficulty level(1~3). Respond according to the [Template] below. **Problem Settings** ▪Focus on core C language concepts ▪Evaluate computational thinking rather than syntax memorization **Level Settings** ▪Level 1: Sequential execution, basic input/output (5-7 lines) ▪Level 2: Include conditionals, loops (7-10 lines) ▪Level 3: Utilize functions, arrays, pointers (10-15 lines) [Template] ### Problem: ▪Real-life scenario problem ▪Algorithm Description: Algorithm explanation ### Parsons Problem Arrange the following code lines in correct order to solve the problem: ```c 1. [Code line 1] 2. [Code line 2] 3. [Code line 3] 4. [Code line 4] .... up to 15 lines maximum ```
Debugging	Provide (hint type) for (improving/fixing errors) in the code below according to the ** Hint Type** ▪Directional hints: Detailed guidance on areas to focus on ▪Tool-based hints: Brief description of tasks to perform ▪Example-based hints: Provide similar example code (Code) (Error message) (Improvements) [Template] ### Problem Analysis ▪Error Type: Compile/Runtime/Logic ▪Location: Line number or function name ▪Symptom Summary: Summarize current situation in 1-2 sentences ### Hints ▪Hint: Explanation using user-selected hint type Prevention Method: How to avoid the same mistakes in the future ▪Good Practices: Things to always check when coding

개념 학습과 프로그래밍 연습 문제, 디버깅 프롬프트의 전반적인 응답 정확성을 높이기 위해 더 상세한 맥락을 포함할 수 있는 지시 사항을 추가하였다. 특히 개념 학습은 초보자가 유의할 사항에 대한 생성을 추가하고 예시 코드를 생성할 때는 마크다운을 활용하여 정확한 코드 양식으로 작성될 수 있도록 템플릿을 개선하였다. 프로그래밍 연습 문제는 전문가의 의견을 반영하여 문제의 난이도를 1~3까지로 설정하고 해당 난이도에 대한 지시 사항도 명확하게 구성하였다. 또한 파슨스 문제를 정확하게 생성할 수 있는 마크다운 코드 양식을 활용하여 프롬프트를 구성하였다. 마지막으로 디버깅의 경우 프로그래밍 시 발생할 수 있는 오류 메시지를 입력할 수 있도록 템플릿을 수정하고 초보 학습자에게 도움이 될 수 있는 오류 위치나 전체적인 개선 사항을 요약해서 생성할 수 있도록 수정하였다.

또한 풍부한 맥락을 제공하되 토큰 사용량을 고려하여 프롬프트가 효율적으로 구성되도록 반복적인 단어나 문장을 생략하고 수정하였다. 또한 토큰 수를 계산하는 서비스를 활용하여 전반적인 프롬프트의 의미를 유지할 수 있는 개조식 문장으로 수정하였다.

4. Second Analysis Results

개선된 PEP를 동일한 전문가 집단에게 직접 시연하고 1주일간 활용 후 적합성과 LLM 응답의 정확도를 평가할 수 있는 동일한 설문에 응답 요청하였다. 12명의 전문가가 모두 응답하였으며 응답 결과는 <Table 6>과 같다.

www.kci.go.kr

Table 6. Results of Second Experts Review

Validity	Agreements	CVR
Role setting	12	1.00
Concept Learning	12	1.00
Programming Exercise	11	0.83
Debugging	11	0.83
Accuracy	M	SD
Concept Learning	4.67	0.47
Programming Exercise	4.00	0.58
Debugging	4.83	0.37

모든 프롬프트의 적합성이 CVR 값 기준(CVR>0.56)을 초과하였으며 PEP의 초안과 비교하여 개선된 것을 확인할 수 있다. 또한 정확도의 평균도 모두 4.00 이상으로 확인되었으며 평균과 표준 편차 모두 PEP 초안과 비교하여 개선되었다. 해당 설문 결과를 종합하여 분석하고 추가적인 인터뷰는 실시하지 않았으며 설문에 포함된 의견 중 해당 프롬프트를 활용할 수 있는 설명 자료가 필요하다는 의견을 반영하여 간단한 학습자용 설명 자료를 추가적으로 작성하고 PEP 개발을 완료하였다.

5. Effectiveness Analysis

설계 기반 연구의 절차에 따라 개발된 PEP의 효과성을 확인하고 프로그래밍 교육을 위한 LLM 활용의 이론을 일반화하기 위해 <Table 7>과 같이 준실험을 구성하였다.

Table 7. Quasi-Experimental Design

G_1	O_1	X_1	O_3
G_2	O_2	X_2	O_4

G1: Experimental Group(n=23)
G2: Control Group(n=23)
O1, O2: Pre-Test Scale for Developing Programming-Oriented Computational Thinking
O3, O4: Post-Test Scale for Developing Programming-Oriented Computational Thinking
X1: Utilizing LLM with PEP throughout C programming
X2: Utilizing LLM without restrictions throughout C programming

매주 3시간씩 15주간의 C 프로그래밍 수업에서 실험 집단은 첫 시간에 PEP의 소개와 함께 학습자용 설명 자료를 제공하고 프롬프트 구조의 수정 없이 개념 학습, 실습, 과제 도우미로 활용하게 하였다. 이에 반해 통제집단은 명시적으로 특정 개념에 대한 설명이나 코드를 직접적으로 생성하는 일반적인 프롬프트를 안내하고 제한 없이 활용할 수 있도록 하였으며 실험집단과 동일하게 수업 중 개념 학습, 실습, 과제 도우미로 활용하도록 안내하였다.

준실험을 통한 결과 해석의 타당도를 높이기 위해 두 집단의 컴퓨팅 사고력을 사전 평가하고 동질성을 확인하였다. 집단 간 차이를 정량적으로 평가하는 통계적 방법 설정을 위해 컴퓨팅 사고력 하위 요인 별 응답 결과를 활용하여 집단별 정규성을 확인할 수 있는 Shapiro-Wilk 검정을 진행하였다. 검증 결과는 <Table 8>과 같다.

Table 8. Results of Shapiro-Wilk Normality Tests

Factor	Group	W	p	Normality
Conceptual Knowledge	Control	0.944	.240	O
	Experi-ment	0.887	.017	X
Algorithmic Thinking	Control	0.893	.022	X
	Experi-ment	0.913	.053	O
Evaluation	Control	0.907	.041	X
	Experi-ment	0.904	.035	X

정규성 검정 결과, 4가지 항목에서 정규분포 가정이 충족되지 않았다($p<.05$). 따라서 방법론적 일관성을 확보하기 위해 모든 하위 요인에 대해 비모수 검정인 Mann-Whitney U 검정을 실시하였다. Mann-Whitney U 검정은 비모수 검정으로 샘플의 수가 작거나 정규 분포를 따르지 않을 때 활용할 수 있다는 장점이 있다. 이에 본 연구의 집단 간 비교에 적합한 방법으로 볼 수 있다[35].

하위 요인별 각 집단의 컴퓨팅 사고력 사전 평가 결과는 <Table 9>와 같다.

Table 9. Results of Computational Thinking Pre-test

Factor	Group	M	SD	Mann-Whitney U	
				z	p
Conceptual Knowledge	Control	2.84	0.91	-0.211	.842
	Experi-ment	2.71	1.05		
Algorithmic Thinking	Control	2.40	0.75	-0.059	.962
	Experi-ment	2.45	0.99		
Evaluation	Control	2.78	0.90	-0.270	.796
	Experi-ment	2.71	0.96		

컴퓨팅 사고력의 하위 요인인 개념적 지식, 알고리즘적 사고, 평가에서 평균, 표준편차 모두 큰 차이를 보이지 않았다. Mann-Whitney U 검정을 통해 집단 간의 차이를 비교한 결과 또한 하위 요인 모두 통계적으로 유의미한 차이가 없는 것으로 나타났다($p>.05$).

집단 별 결과의 4분위 값 위치와 분포에 대해서 다각도로 분석하기 위해 (Fig. 2)와 같이 박스 플롯을 사용하여 결과를 시각화하였다.

www.kci.go.kr

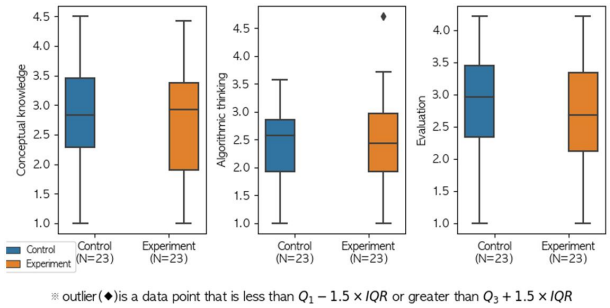

Fig. 2. Pre-test Visualization

최댓값의 차이는 있지만 사분위 범위는 집단에 따른 차이가 크게 나타나지 않았다. 또한 실험 집단과 통제 집단 동일하게 모든 하위 요인의 결과가 1.0에서 3.5 이상의 넓은 범위에 분포하고 있다. 이를 통해 두 집단은 전체적으로 유사한 컴퓨팅 사고력 역량을 지니고 있으며 학생들 간의 컴퓨팅 사고력 역량 편차가 크게 나타남을 확인할 수 있다. 각 하위 요인에서 4.0 이상의 높은 점수로 응답한 학생의 특성을 파악하기 위해 개별 면담 진행하였으며 해당 학생들은 C프로그래밍의 경험은 없지만 모두 스크래치나 엔트리와 같은 비주얼 중심의 프로그래밍 경험을 가지고 있었으며 이를 통해 프로그래밍에 대한 일부 개념적 지식과 컴퓨팅 사고력을 갖추고 있다고 스스로 평가함을 확인할 수 있었다.

실험 처치를 적용한 15주 수업이 종료된 후 사후 평가를 위해서 동일한 컴퓨팅 사고력 검사지를 활용하였다.

체계적인 분석을 위해서 Mann-Whitney U 검정을 이용해서 집단 간 차이를 분석하였다. 또한 본 연구는 컴퓨팅 사고력의 3개 하위 요인에 대해 실험집단과 통제집단의 독립적인 통계 검정을 수행한다. 따라서 다중비교로 인한 오류 누적을 보정하고자 Bonferroni 보정을 적용하여 유의수준(α=0.05/3=0.0167)을 조정하여 평가하였다.

추가적으로 집단 간 통계적 유의성뿐만 아니라 효과 크기 분석을 통해 실질적 효과성을 평가하였다. 효과 크기 분석을 위한 방법으로는 비모수적 효과 크기인 Cliff's Delta(δ)를 산출하였다. 효과 크기의 기준은 Romano 외(2006)의 해석 기준에 따라 |δ|<0.147은 무시할 수준, 0.147≤|δ|<0.330은 작은 효과, 0.330≤|δ|<0.474는 중간 효과, |δ|≥0.474는 큰 효과로 해석한다[36]. 마지막으로 95%의 신뢰구간(CI)을 부트스트래핑(재표집 5,000회)로 추정하여 효과의 안정성을 추가적으로 판단하였다. 집단 간 사후 평가의 결과는 <Table 10>과 같다.

Table 10. Results of Computational Thinking Post-test

Factor	Group	M	SD	Mann-Whitney U	
				z	P[1)]
Conceptual Knowledge	Control	3.54	0.54	-1.268	.207
	Experiment	3.67	0.62		
Algorithmic Thinking	Control	3.20	0.64	-2.582	.010*
	Experiment	3.52	0.68		
Evaluation	Control	3.37	0.71	-2.300	.022
	Experiment	3.60	0.62		

Factor	Cliff's δ[2)]	95% CI[3)]	Effect Size
Conceptual Knowledge	0.22	[-0.12, 0.56]	Small
Algorithmic Thinking	0.45	[0.14, 0.74]	Medium-large
Evaluation	0.41	[0.07, 0.71]	Medium

1) *p < .0167 (Bonferroni corrected: α=0.05/3)
2) |δ| < 0.147 (negligible), 0.147 ≤ |δ| < 0.330 (small), 0.330 ≤ |δ| < 0.474 (medium), |δ| ≥ 0.474 (large)
3) 95% CI=Confidence interval estimated using bootstrapping with 5,000 resamples.

개념적 지식의 경우 실험집단의 평균(M=3.67, SD=0.62)이 통제집단의 평균(M=3.54, SD=0.54)에 비해 소폭 높게 나타났으나, Mann-Whitney U 검정 결과 두 집단 간의 차이는 통계적으로 유의하지 않았다(z=-1.268, p=.207). 효과 크기 분석 결과, Cliff's Delta는 δ=0.22 (95% CI: -0.12, 0.56)로 작은 크기의 효과를 나타냈다. 95% 신뢰구간 추정 결과를 살펴보면 -0.12에서 0.56까지 분포하며 0을 포함하고 있기 때문에 실험집단이 통제집단보다 높을 수도 있지만, 반대로 낮을 가능성도 있음을 의미하며, 효과의 방향성조차 불확실함을 시사하였다.

알고리즘적 사고의 경우, 실험집단의 평균(M=3.52, SD=0.68)이 통제집단(M=3.20, SD=0.64)보다 현저히 높게 나타났으며, Mann-Whitney U 검정 결과, 두 집단의 차이가 Bonferroni 보정된 유의수준에서 통계적으로 유의하게 나타났다(z=-2.582, p=.010). 효과 크기 분석 결과, Cliff's Delta는 δ=0.45 (95% CI: 0.14, 0.74)로 중간에서 큰 효과 크기의 경계에 위치하였다. 95% 신뢰구간이 0.14에서 0.74 사이에 분포하고 0을 포함하지 않으며 신뢰구간의 하한(0.14)조차 작은 효과 크기의 범위에 있어, 효과의 존재와 방향성이 안정적으로 확인되었다.

평가 요인의 경우, 실험집단의 평균(M=3.60, SD=0.62)이 통제집단(M=3.37, SD=0.71)보다 높게 나타났다. Mann-Whitney U 검정 결과 p=.022로, 일반적인 유의수

www.kci.go.kr

준(α=.05)에서는 유의하였으나 Bonferroni 보정된 유의수준(α=.0167)에는 미치지 못하였다(z=-2.300, p=.022). Cliff's Delta는 δ=0.41 (95% CI: 0.07, 0.71)로 중간 크기의 실질적 효과를 나타내었으며 95% 신뢰구간이 0.07에서 0.71 사이에 분포하여 효과의 방향성은 일관되게 나타났다. p=.022는 Bonferroni 보정 기준(α=.0167)에 약간 미치지 못하지만, 중간 크기의 효과 크기(δ=0.41)와 0을 포함하지 않는 신뢰구간은 PEP가 평가 역량 향상에도 실질적으로 긍정적인 영향을 미칠 가능성을 시사하였다.

사전 검사와 마찬가지로 그룹 간 분포 비교를 위해 사후 검사 결과를 (Fig. 3)과 같이 시각화하였다.

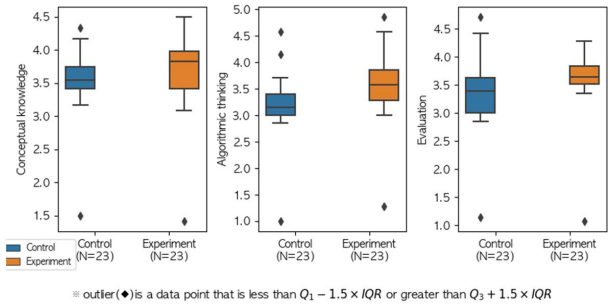

Fig. 3. Post-test Visualization

두 집단 모두 3가지 하위 요인 결과에서 최솟값을 제외하면 대부분의 학생이 약 3에서 4.5 사이의 결과에 분포하고 있음을 확인할 수 있다. 이는 사전 검사에 비해 모든 결과가 비교적 좁은 범위에 군집을 이루고 있으며 프로그래밍 수업은 학생 간의 컴퓨팅 사고력 역량 편차를 해소하는 데 도움을 주고 전체적인 컴퓨팅 사고력 향상에 긍정적인 영향을 준 것으로 해석할 수 있다.

개념적 지식 하위 요인의 최댓값은 실험집단이 더 높게 나타났지만, 최솟값은 더 낮게 나타났다. 또한 실험 집단의 개별 학생 분포와 사분위 범위가 통제집단에 비해 다소 넓게 나타났지만 큰 차이를 보이지 않았다.

알고리즘적 사고의 경우 실험 집단이 3.29 ~ 4.0에 가장 많이 분포하고 있는 데 비해 통제집단은 3.0 ~ 3.5 사이에 가장 많이 분포하고 있으며 사분위 범위 또한 실험집단이 상대적으로 높은 범위에 위치하고 있어 집단 간의 차이를 확인할 수 있다.

평가 하위 요인은 통제집단의 최댓값과 최솟값이 더 높게 나타났다. 특히 실험군에서는 관찰되지 않았던 4.3을 초과하는 최댓값이 통제집단에서 나타남을 확인할 수 있었다. 하지만 통제 집단은 대부분의 학생이 2.8~3.5 범위에 집중되어 있는데 반해 실험 집단은 주로 3.4~4.0 사이의 좁은 범위에 집중되어 있음을 확인할 수 있다. 이를 통해 평가 요인은 실험집단이 통제 집단에 비해 더 일관되고 높은 성과를 보이고 있음을 확인할 수 있다.

V. Conclusions

본 연구는 설계 기반 연구 방법론을 통해 LLM에 적용할 수 있는 프로그래밍 교육용 프롬프트(PEP)를 개발하고 그 효과성을 검증하였다. 설계, 실행, 분석, 재설계의 반복적 순환 과정을 통해 교육 현장에 실질적으로 적용 가능한 설계 원칙과 이론적 기여를 다음과 도출할 수 있다.

1. Design Principles and Theoretical Contributions

본 연구는 설계 기반 연구의 핵심 결론으로 프로그래밍 교육을 위한 LLM 프롬프트 설계 원칙을 도출하였다.

첫째, LLM의 교육적 활용을 위해서는 명확한 페르소나와 맥락 설정을 통한 역할 부여가 필요하다. 전문가 검증 결과, '프로그래밍 교육 전문가' 페르소나는 LLM의 응답을 교육적 맥락으로 유도하는 핵심 기제로 분석되었다. 또한 일부 전문가는 학습 목표와 초보 학습자 배려 등 더 상세한 교육적 맥락 추가를 제안하였으며, 이를 반영하여 초보자를 위한 구체적 지침과 금지 사항을 추가하여 교육적 경계를 명확히 설정하였다. 이러한 설계는 LLM이 단순 코드를 생성하는 도구가 아닌, 프로그래밍 학습을 단계적으로 안내하는 지원 도구로 작동하도록 유도하였다.

둘째, 템플릿 패턴을 활용한 응답 구조화는 LLM의 확률적 생성 특성을 효과적으로 제어할 수 있다. 프로그래밍 연습(파슨스 문제) 프롬프트에 대한 전문가 검증 과정에서 응답 구조화의 중요성이 특히 강조되었다. 초기 프롬프트는 코드 라인 생성의 불일관성으로 인해 학습자 혼란이 우려되었으며, 이를 해결하기 위해 마크다운 코드 블록을 명시적으로 지정하고, 난이도를 1~3단계로 명확히 구조화하였으며, 교육적 목표를 구체적으로 명시하였다. 구조화된 파슨스 문제는 학습자가 알고리즘의 논리적 흐름을 체계적으로 이해하도록 지원하여, 실험집단의 알고리즘적 사고력 향상에 직접 기여한 것으로 분석된다.

셋째, 직접적 답안 제시를 지양하고 다양한 힌트 유형을 통한 차별화된 비계를 제공해야 한다. 전문가들은 LLM이 직접적 해결책 대신 단계적 힌트를 제공하는 접근이 교육적으로 매우 효과적이라고 평가하였다. 추가 의견으로 코드와 에러 메시지를 함께 입력하는 방법과 단순 오류 수정을 넘어선 코드 개선 방향을 생성하는 프롬프트를 포함하도록 제안하였다. 이를 반영하여 오류 분석, 예방 방법, 모

www.kci.go.kr

범 사례를 제공할 수 있도록 구조를 개선하였으며 이는 학습자의 메타인지적 사고를 촉진하고, 자신의 코드를 비판적으로 검토하며 개선점을 찾는 능력을 향상시킬 수 있었던 것으로 해석된다. 그 결과 평가 하위 요인에서 실험 집단은 통제 집단과 비교하여 중간 크기의 실질적 효과와 일관된 방향성을 보여주었다.

이러한 설계 원칙은 기존 연구의 한계를 극복하는 새로운 이론적 프레임워크를 제시한다. 프로그래밍과 LLM을 중심으로 진행된 선행 연구들은 주로 프로그래밍의 생산성을 높이기 위한 방향을 제안하는 형태로 이루어졌다[22-24]. 교육에 초점을 맞춘 연구에서도 주로 LLM의 적용 가능성을 확인하거나 API를 활용하여 개별 서비스 혹은 인터페이스를 개발하는 수준에 머물렀다[13,14,27]. 본 연구는 이러한 격차를 해소하기 위해 교육적 관점을 중심으로 프로그래밍 교육 프롬프트를 체계적으로 개발하고 실제 교육 현장에 적용하여 효과성을 증명하였다. 특히 프롬프트는 LLM 모델에 종속되지 않고 사용량 제한에 비교적 자유로우며, 교육 환경에 적용하기 위한 유지 보수 측면에서 추가적인 비용이 필요하지 않다는 점에서 장점을 갖는다[28,29]. 또한 자연어를 중심으로 구성된 프롬프트는 다양한 교수학습 환경에 맞게 손쉽게 수정되고 재창조될 수 있기 때문에 교육적 파급력이 높다. 이는 모든 교육 환경에서 제한 없이 지속 가능한 교육을 지원하기 위한 프롬프트 중심의 연구가 필요함을 시사한다[21,34].

2. Practical Implications

본 연구의 결과는 교사, 학생, 교육기관, 교육 정책 수립기관 등 다양한 이해 관계자에게 실질적인 시사점을 제공한다. 교사는 본 연구의 결과물인 프롬프트를 활용하여 자유롭게 수정하며 지속 가능한 프로그래밍 교육 교수학습 환경을 구성할 수 있다. 학생은 프로그래밍 학습이 이루어지는 모든 상황에서 프롬프트를 적용하고 LLM을 튜터로 활용할 수 있다.

교육기관은 프로그래밍 교육뿐만 아니라 다양한 교과에서 교육용 프롬프트를 실제적으로 적용하고 교육의 질을 높일 수 있는 방안을 연구해야 한다. 정책 수립기관은 LLM의 교육적 활용 가능성을 기반으로 학생 및 교사가 인공지능을 활용할 수 있는 재정적 지원과 정책적 기반을 수립해야 한다.

본 연구에 참여한 프로그래밍 교육 전문가 12명은 단순한 검증을 넘어 설계의 공동 창조자 역할을 수행하였다. 전문가들은 프롬프트의 교육적 적합성뿐만 아니라 LLM이 생성하는 답변의 무작위성을 고려하여 가장 최적의 답변을 제공할 수 있는 방향에 대한 의견을 제시하였다. 이는 교육을 위한 목적의 LLM 관련 연구에서 교육 전문가를 비롯한 현장의 의견을 수렴할 수 있는 다학제적 구성원의 참여가 필수적임을 보여준다[25,26].

3. Limitations and Future Research Direction

본 연구는 몇 가지 한계를 가진다. 프롬프트 효과성 검증을 위한 표본 크기가 상대적으로 작고, 대학생으로 한정되어 있다. 또한 효과성 검증을 위해 학습자의 자기 보고식 설문 활용하였으며 개념적 지식 요인에서 유의미한 차이를 발견하지 못한 점도 추가 연구가 필요한 부분이다.

향후 연구에서는 더 다양한 대상과 더 큰 표본으로 확대하여 대상의 특성에 따른 프롬프트의 유형과 효과성에 대한 심도 있는 연구가 필요하다. 또한 프로그래밍 교육을 위한 프롬프트를 포함하여 면밀한 교수학습 모델을 개발하고 적용할 필요가 있다. 다양한 프로그래밍 언어에 대한 프롬프트 확장, 장기적 학습 효과 추적 연구, 오픈소스 LLM 모델에서의 적용 가능성 검증도 중요한 과제이다.

본 연구는 설계 기반 연구 방법을 통해 LLM을 활용한 프로그래밍 교육이 단순한 기술 도입이 아닌, 교육적 설계를 통해 구현되어야 함을 입증하였다. PEP는 프롬프트가 '어떻게' LLM이 교육적 응답을 생성하도록 유도하는지, 그리고 '왜' 이러한 접근이 전통적 프로그래밍 교육의 한계를 극복할 수 있는지에 대한 방향성을 제시하였다.

본 연구에서 도출된 설계 원칙과 실증적 증거는 LLM이 컴퓨팅 사고력 신장을 위한 효과적인 교육 도구로 활용될 수 있음을 보여준다. 특히 프롬프트 중심의 접근은 모든 교육 환경에서 제한 없이 활용 가능한 지속 가능한 교육적 이점을 제공한다. 이는 디지털 시대의 필수 역량인 컴퓨팅 사고력의 향상을 위한 효과적인 교육적 도구 및 도구의 기저로서 다양한 연구와 현장에 기여할 것이다.

REFERENCES

[1] P. J. Denning, “Computing is a natural science,” Commun. ACM, vol. 50, no. 7, pp. 13-18, Jul. 2007, DOI:10.1145/1272516.1272529.

[2] J. M. Wing, “Computational thinking and thinking about computing,” Philosophical Transactions of the Royal SocietyA: Mathematical, Physical and Engineering Sciences, vol. 366, no. 1881, pp.3717-3725, Jul. 2008, DOI:10.1098/rsta.2008.0118.

[3] D. Weintrop et al., “DefiningComputational Thinking for Mathematics and Science Classrooms,” J Sci EducTechnol, vol. 25, no. 1, pp. 127-147, Feb. 2016, DOI:10.1007/s10956-015-

www.kci.go.kr

9581-5.

[4] S. Jeon and Y. Lee, “A meta analysis of programming education effects according to learning activity themes,” The Journal of Korean Association of Computer Education, vol. 19, no. 2, pp.21-29, 2016, DOI:10.32431/kace.2016.19.2.003.

[5] H. Gao, Z. Qiu, D. Wu, and L. Gao, “Research and Reflection on Teaching of C Programming Language Design,” in Intelligent Computation in Big Data Era, Springer, 2015, pp. 370-377. June. 2015, doi: 10.1007/978-3-662-46248-5_45

[6] C. Changpetch, P. Panjaburee, and N. Srisawasdi, “A comparison of pre-service teachers’ variable misconceptions in various computer-programming preferences: findings to teacher education course,” J. Comput. Educ., vol. 9, no. 2, pp. 149-172, Jun. 2022, DOI:10.1007/s40692-021-00200-0.

[7] G. S. Sheikh and N. Islam, “A qualitative study of major programming languages: teaching programming languages to computer science students,” Journal of Information & CommunicationTechnology (JICT), vol. 10, no. 1, p. 11, 2016.

[8] N. Weinman, A. Fox, and M. A. Hearst, “Improving Instruction of Programming Patterns with Faded Parsons Problems,” in Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems,Yokohama Japan: ACM, pp. 1-4, May 2021, DOI:10.1145/3411764.3445228.

[9] A. Gomes and A. J. Mendes, “An environment to improve programming education,” in Proceedings of the 2007international conference on Computer systems and technologies - CompSysTech ’07, Bulgaria: ACM Press,2007, p. 1. DOI:10.1145/1330598.1330691.

[10] C. C. Selby, “Relationships: computational thinking, pedagogy of programming, and Bloom’s Taxonomy,” in Proceedings ofthe Workshop in Primary and Secondary Computing Education, London UnitedKingdom: ACM, pp. 80-87, Nov. 2015, DOI:10.1145/2818314.2818315.

[11] Z. Yinnan and L. Chaosheng, “Training for computational thinking capability on programming language teaching,” in 20127th International Conference on Computer Science & Education (ICCSE),Melbourne, Australia: IEEE, pp. 1804-1809, Jul. 2012, DOI:10.1109/ICCSE.2012.6295420.

[12] M. Noone and A. Mooney, “Visual and textual programming languages: a systematic review of the literature,” J. Comput.Educ., vol. 5, no. 2, pp. 149-174, Jun. 2018, DOI:10.1007/s40692-018-0101-5.

[13] J. Leinonen et al., “Comparing CodeExplanations Created by Students and Large Language Models,” in Proceedings of the 2023 Conference on Innovation and Technology in Computer ScienceEducation V. 1, Turku Finland: ACM, pp. 124-130, Jun. 2023, DOI:10.1145/3587102.3588785.

[14] R. Xiao, X. Hou, and J. Stamper, “Exploring How Multiple Levels of GPT-Generated Programming Hints Support or Disappoint Novices,” in Extended Abstracts of the CHI Conference on Human Factors inComputing Systems, pp. 1-10, May 2024, DOI:10.1145/3613905.3650937.

[15] M. Piteira and C. Costa, “Learning computer programming: study of difficulties in learning programming,” in Proceedings of the 2013 International Conference on Information Systems and Design ofCommunication, Lisboa Portugal: ACM, pp. 75-80, Jul. 2013, DOI:10.1145/2503859.2503871.

[16] Z. Gao, S. Heckman, and C. Lynch, “Who Uses Office Hours?: A Comparison of In-Person and Virtual Office Hours Utilization,” in Proceedings of the 53rd ACM Technical Symposium on Computer ScienceEducation, Providence RI USA: ACM, pp. 300-306, Feb. 2022, DOI:10.1145/3478431.3499334.

[17] T. Jenkins, “Teaching programming-A journey from teacher to motivator,” in The 2nd Annual Conference of the LSTN Centerfor Information and Computer Science, Citeseer, 2001.

[18] D. Gupta, “What is a good first programming language?,” XRDS, vol. 10, no. 4, pp. 7-7, Aug. 2004, DOI:10.1145/1027313.1027320.

[19] X. Hou, B. J. Ericson, and X. Wang, “Integrating Personalized Parsons Problems with Multi-Level Textual Explanations to Scaffold Code Writing,” in Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2, pp.1686-1687, Mar. 2024, DOI:10.1145/3626253.3635606.

[20] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye, “A review on generative adversarial networks: Algorithms, theory, and applications,” IEEE transactions on knowledge and data engineering, vol.35, no. 4, pp. 3313-3332, 2021.

[21] W. Hariri, “Unlocking the Potential of ChatGPT: A Comprehensive Exploration of its Applications, Advantages, Limitations, and Future Directions in Natural Language Processing,” Apr. 12, 2023, arXiv:2304.02017.

[22] C. Liu et al., “Improving ChatGPTPrompt for Code Generation,” May 15, 2023, arXiv:2305.08360.

[23] H. Tian et al., “Is ChatGPT theUltimate Programming Assistant -- How far is it?,” Aug. 31, 2023, arXiv:2304.11938.

[24] S. Biswas, “Role of ChatGPT in Computer Programming.: ChatGPT in Computer Programming.,” Mesopotamian Journal ofComputer Science, vol. 2023, pp. 8-16, 2023.

[25] L. Avila-Chauvet, D. Mejía, and C. O. Acosta Quiroz, “Chatgpt as a support tool for online behavioral task programming,” Available at SSRN 4329020, 2023.

[26] J. Cao, M. Li, M. Wen, and S. Cheung, “A study on Prompt Design, Advantages and Limitations of ChatGPT for Deep Learning Program Repair,” Apr. 17, 2023, arXiv:2304.08191.

[27] A. Hellas, J. Leinonen, S. Sarsa, C. Koutcheme, L. Kujanpää, and J. Sorva, “Exploring the Responses of Large Language Models to Beginner Programmers’ Help Requests,” in Proceedings ofthe 2023 ACM Conference on International

www.kci.go.kr

Computing Education Research V.1,Chicago IL USA: ACM, pp. 93-105, Aug. 2023, DOI:10.1145/3568813.3600139.

[28] M. Kazemitabaar et al., "CodeAid:Evaluating a Classroom Deployment of an LLM-based Programming Assistant thatBalances Student and Educator Needs," in Proceedings of the CHI Conferenceon Human Factors in Computing Systems, pp. 1-20, May 2024, DOI:10.1145/3613904.3642773.

[29] J. Prather et al., "'It's Weird Thatit Knows What I Want': Usability and Interactions with Copilot for NoviceProgrammers," ACM Trans. Comput.-Hum. Interact., vol. 31, no. 1, pp.1-31, Feb. 2024, DOI:10.1145/3617367.

[30] F. Wang and M. J. Hannafin, "Design-based research and technology-enhanced learning environments," ETR&D, vol.53, no. 4, pp. 5-23, Dec. 2005, DOI:10.1007/BF02504682.

[31] The Design-Based Research Collective, "Design-Based Research: An Emerging Paradigm for Educational Inquiry," EducationalResearcher, vol. 32, no. 1, pp. 5-8, Jan. 2003, DOI:10.3102/0013189X032001005.

[32] C. H. Lawshe, "A Quantitative approach to content validity," Personnel Psychology, vol. 28, no. 4, pp. 563-575, Dec. 1975, DOI:10.1111/j.1744-6570.1975.tb01393.x.

[33] S. Kılıç, S. Gökoğlu, and M. Öztürk, "A Valid and Reliable Scale for Developing Programming-Oriented Computational Thinking," Journal of Educational Computing Research, vol. 59, no. 2, pp. 257-286, Apr. 2021, DOI:10.1177/0735633120964402.

[34] J. White et al., "A Prompt PatternCatalog to Enhance Prompt Engineering with ChatGPT," Feb. 21, 2023, arXiv:2302.11382.

[35] G. D. Ruxton, "The unequal variance t-test is an underused alternative to Student's t-test and the Mann-Whitney U test," BehavioralEcology, vol. 17, no. 4, pp. 688-690, 2006.

[36] J. Romano, J. D. Kromrey, J. Coraggio, and J. Skowronek, "Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen'sd for evaluating group differences on the NSSE and other surveys," in annual meeting of the Florida Association of Institutional Research, 2006.

Authors

Seul-ki Kim received the B.S degrees in Elementary School Education and M.S. degrees in Convergence Education from Gyeongin National University of Education, Ph. D. degrees in Computer Science Education from Korea National University of Education. He is currently a teacher in the elementary school, of gyeonggi-do Office of Education, since 2008. He is interested in Computer Science Education, Programming Education, Data Literacy and Artificial Intelligence Education.

www.kci.go.kr