



# 고등학교 정보 과목의 프로그래밍 영역 평가 문항 유형과 프로그래밍 요소 기반 난이도 측정 방법 개발\*

## Development of Evaluation Question Types and Difficulty Level Measurement Methods Based on Programming Elements for the Programming Area of High School Informatics Subject

박찬정<sup>†</sup> · 현정석<sup>††</sup>

Chan Jung Park<sup>†</sup> · Jung Suk Hyun<sup>††</sup>

### 요약

2022 개정 교육과정에서는 언어, 수리, 디지털을 초·중등 학생의 기본 소양으로 설정하였다. 또한, 인공지능 및 소프트웨어 교육의 중요성이 커짐에 따라, 학교 현장에서는 디지털 소양 함양 교육과 정보 교육이 강화되고 있다. 한편, 정보 교육에서는 2015 개정 교육과정에서부터 컴퓨팅 사고력을 신장시키고 문제해결력을 높이기 위해 효과적인 컴퓨터 프로그래밍 교육 방법에 관한 연구가 활발히 진행되었다. 본 연구는 컴퓨팅 사고력 요소 중 자동화에 초점을 두고 파이썬 프로그래밍 언어를 중심으로 고등학생을 위한 다양한 프로그래밍 문제 유형을 개발하는 데 목적을 두었다. 또한 본 연구는 문제의 난이도를 결정하기 위해 프로그래밍 요소를 기반으로 한 방법을 제시하여 다양한 유형의 문제 50개를 생성형 AI로 생성한 후, 그 중 20문제를 선정하고 보완하여 정보 교사에게 해결하게 하였다. 이후, 응답 결과를 분석하여 문제 유형과 난이도 적합성을 평가하였다. 연구 결과물은 고등학교 지필 평가는 물론 향후 수능에 정보 교과가 포함된다면, 수능에서 프로그래밍 역량 평가에 활용될 수 있다.

주제어 컴퓨터 교육 평가, 고등학교 정보, 프로그래밍 교육, 객관식 문항 개발, 생성형 AI

### ABSTRACT

In the 2022 revised curriculum, language, numeracy, and digital literacy were established as fundamental competencies for elementary and secondary students. With the growing importance of artificial intelligence and software education, schools are increasingly focusing on digital literacy and Informatics education. Meanwhile, since the 2015 revised curriculum, there has been active research on effective computer programming teaching methods aimed at enhancing computational thinking and problem-solving skills. This paper focuses on the automation aspect of computational thinking and aims to develop various types of programming problems for high school students, centered on the Python programming language. In addition, this research proposed a method based on programming elements to determine the difficulty of the problem, and 50 different types of problems were generated using generative AI. From these, 20 problems were selected and refined, then given to Informatics teachers to solve. The response results were analyzed to evaluate the appropriateness of the problem types and difficulty levels. The outcomes of this research could be applied in high school written exams and, if the informatics subject is included in future college entrance exams, they could also be used to assess programming competencies.

Keywords Computer Education Assessment, High School Informatics, Programming Education, Multiple-Choice Question Development, Generative AI

†중신회원 제주대학교 사범대학 컴퓨터교육과 교수

††중신회원 제주대학교 경상대학 경영정보학과 교수 (교신저자)

논문투고 2024년 08월 01일

심사완료 2024년 09월 26일

게재확정 2024년 10월 02일

발행일자 2024년 10월 10일

\* IRB 승인번호 : JJNU-IRB-2024-063

## 1. 서론

현대 사회에서 소프트웨어(SW)와 인공지능(AI)이 차지하는 중요성이 더욱더 커지고 있으며, 여러 국가는 이에 맞춰 교육과정을 개편하고 있다. 미국은 초중등 학생을 위한 인공지능 교육 기준안(AI4K12)을 개발했고, 핀란드는 Elements of AI 프로그램을 도입했다. 중국은 차세대 인공지능 발전 계획과 고등교육 인공지능 혁신 행동 계획을 발표하며 프로그래밍 교육을 의무화했다[1].

한국에서도 2015 개정 교육과정에서 초등 17시간, 중등 34시간의 정보 교육을 의무화했고, 2022 개정 교육과정에서는 초등 34시간, 중등 68시간 이상 필수로 이수해야 한다고 발표했다. 이는 창의 융합형 인재 양성을 위한 SW·AI 교육의 기반을 마련하기 위한 정책과 연구 노력의 일환이다. 그러나 일부 연구에 의하면, 교육 현장에서 정부가 기대하는 소프트웨어 교육 효과는 미미하게 나타났다[2]. 프로그래밍 교육은 현재 문제 해결, 실습 기반 학습, 프로젝트 기반 학습에 중점을 두고 있으며, 그 결과 ‘코드 작성 능력’에 집중하고 있다. 프로그래밍 수업에서 교사들은 시간 제약과 대규모 학생 관리로 인해 온라인 채점 프로그램을 사용하거나 부분적 코드 작성을 진행한다. 이로 인해 학생들이 문제해결에 적극적으로 참여하기보다는 모범 답안을 기다리는 경우가 많다는 교사의 지적이 있다.

과거와 다르게 거의 모든 학생이 프로그래밍 교육을 받게 되면서, 학생들이 부담 없이 프로그래밍에 접근할 수 있는 새로운 교육 방법과 교육평가의 필요성이 제기되고 있다. 이에 따라 다양한 교수·학습방법에 관한 연구가 활발히 진행되고 있다[3-8]. 한편, 프로그래밍 언어 교육에서는 수행평가에 주로 프로그램 작성 중심의 프로젝트 기반 학습(PBL)이 이루어지고 있고, 중간시험과 기말시험에서는 주로 객관식 문항을 통해 평가가 이루어지고 있다[9]. 또한, 많은 연구에서 프로그래밍에 관한 문제해결 방식을 백준(www.acmicpc.net)과 KOISTUDY(koistudy.net)와 같은 온라인 저지 기반의 코딩 프로그래밍 문제 해결 방식이 이루어지고 있고[10, 11], 오래되었지만 2013년까지의 수능(직업탐구)의 프로그래밍 과목에서 프로그램 이해 문제(순서도 결과 이해)와 프로그램 작성 문제(빈칸 작성, 코드 순서 맞추기)가 다루어졌다[12]. 하지만 컴퓨터 프로그래밍 영역에서의 문제 유형을 체계적으로 제안한 연구는 드물다.

본 연구는 컴퓨터 프로그래밍 언어를 영어와 같은 일반 언어의 관점에서 바라보고, 평가 문항 유형과 난이도를 측정할 방법을 개발하는 데 목적을 두었다. 최근 한 연구에서는 교과 간 연계성 방안으로 영어와 프로그래밍 교육을 통합하여 언어 통합이 프로그래밍 교육에 긍정적 효과가 있음을 밝혔다. 또한, Python은 영어 어휘와 문장 습득에 효과적이라고 밝혔다[13]. 컴퓨터 프로그래밍 언어가 대상만 인간이 아닌 컴퓨터일 뿐 특정 문법을 가지고 있고, 상호 간 의사소통 도구이며, 언어를 배우는데 학습과 연습이 필요한 공통점이 존재한다. 또한, 두 언어를 구사하는데 모두 사고력이 요구된다. 영어는 오랜 역사를 가지며 고등학생 대상으로 이미

공인된 여러 유형의 시험이 존재하지만 프로그래밍은 그렇지 못하다. 영어 능력을 평가하기 위해 문법, 독해 등 다양한 평가 유형이 존재하는 것처럼, 프로그래밍 언어 능력에 대한 평가 유형도 더 다양하게 개발될 필요가 있다. 본 연구는 일반 언어 중 영어를 예로 들어 다양한 평가 유형을 분석하고, 그 분석을 바탕으로 프로그래밍 언어 능력에 대한 평가 유형 개발을 위한 지침을 제공한다.

본 연구는 영어를 모국어로 하지 않는 학생의 영어 능력을 평가하기 위해 영어권 대학 입시에 사용되는 토플(TOEFL, test of english as a foreign language) 시험의 질문 유형을 참고하였다. 이를 바탕으로 프로그래밍 언어 평가에 적용할 수 있는 유형을 모색한 연구[14]를 참고하여 문제의 유형을 다음과 같이 정의한다. 첫째, 구문 기반 질문이다. 주어진 코드에서 기본 프로그래밍 개념, 구문 또는 기능을 식별하도록 요구하는 질문이 포함된다. 둘째, 의미 기반 질문이다. 함수의 결과 예측과 코드의 논리 또는 출력을 예측하는 추론 질문과 변수값 추측, 함수 의미 이해를 묻는 참조 질문을 들 수 있다. 셋째, 요약 질문은 코드에서 주요 기능이나 개념을 식별하고, 정보를 조직하거나 요약을 완성하게 하여 프로그램의 전체 구조와 목적을 이해하는 능력을 평가한다.

문항의 난이도 측정을 위한 기존의 연구에는 정답률 측정[15], 문항반응이론에 의한 측정[15, 16], 문제해결시간을 기반으로 한 측정[17] 등이 있다. 본 연구는 객관식과 주관식을 모두 다루고 있어, 문항의 난이도 측정을 위해서는 소프트웨어 공학 영역의 소프트웨어 척도(metric) 측정 방법에서 프로그래밍 요소를 기반으로 한 사이클로매틱(cyclomatic) 수[18]와 Halstead의 척도[19]를 기반으로 하였다. 이들 두 가지 방법은 소프트웨어의 복잡도를 계산하는 정량적인 척도를 제공한다.

한편, 교육평가와 관련하여 인공지능을 활용하여 학생들의 프로그램을 자동 채점하고 피드백하는 앱 개발 연구가 늘고 있다[20-22]. 또한 최근 생성형 AI 기반의 다양한 응용 영역이 증가하는 추세이다[23-25]. 하지만, 평가 문항 생성에 관한 연구는 적으며, 특히 정보 교과의 프로그래밍 영역을 위한 시스템을 다룬 연구가 드물다. 본 연구에서는 교육 평가를 위해 문제 유형과 난이도 적용 방법을 개발한 후, 생성형 AI를 활용하여 다양한 프로그래밍 문제를 직접 생성하고, 생성된 문제가 고등학생에게 적합한지 검증한다.

이를 위해 본 연구에서는 고등학교 정보 교육과정에서 내용 요소를 추출하고, 연구자 외 여러 관점에서 다양한 문제의 생성을 위해 생성형 AI(ChatGPT 4o)를 도구로 선택하였다. 문제를 생성할 때 프롬프트의 역할이 중요하기 때문에 프롬프트에 포함시켜야 할 문제의 유형과 프로그래밍 요소 기반 난이도 기준을 설정하였다[26]. 또한, 생성된 문제를 점검하여 20개의 문항을 수정·보완한 후, 정보 교사에게는 객관식으로 문제를 해결하게 하였다. 문제 해결 후, 난이도를 평가하도록 하여 연구에서 측정한 난이도의 적절성을 검증하였다. 본 연구는 문제 유형을 개발하기 위해서 TOEFL

시험[27, 28]에 포함된 문제 유형을 분석하여 현재 프로그래밍 영역의 문제에는 드문 새로운 문제 유형을 정의하였다.

연구 목적을 이루기 위한 본 논문의 구성은 다음과 같다. 2장에서는 영어에서 다루는 문제 유형을 분석하고, 난이도 평가를 위한 이론적 배경을 다룬다. 3장에서는 파이썬 프로그래밍 언어의 내용 요소와 문제 유형을 정의한다. 4장에서는 문제 출제를 통한 생성 문제의 적합성을 검토하고 5장에서 결론을 맺는다.

## 2. 연구 배경

### 2.1 TOEFL의 문제 유형

TOEFL은 영어가 모국어인 아닌 사람들의 영어 능력을 평가하는 시험으로, 미국과 캐나다의 대학 입학 과정에 사용된다. TOEFL iBT는 네 가지 영역으로 구성되어 있다[27]. 문제 유형을 살펴보면 읽기(reading) 영역은 지문의 명확한 정보를 묻는 사실 정보(factual information) 문제, 제시되지 않거나 틀린 정보를 고르는 거짓 정보(negative factual information) 문제, 명시되지 않은 정보를 추론(inference)하는 문제, 저자의 의도나 목적을 묻는 문제, 특정 단어의 의미를 묻는 문제, 특정 단어나 구가 가리키는 대상(reference)을 묻는 문제, 복잡한 문장을 단순화(simplification)한 문장을 고르는 문제, 지문 내의 적절한 위치에 문장을 삽입하는 문제, 요약문 작성이나 표 채우기 문제가 있다[28].

쓰기(writing) 영역은 주어진 주제에 대해 글을 쓰는 능력을 평가한다. 읽기 자료와 강의를 듣고 이를 바탕으로 에세이를 작성하며, 주어진 주제에 대해 자신의 의견을 논리적으로 전개하는 에세이를 작성한다. 말하기(speaking) 영역은 주어진 주제에 대해 말하는 능력을 평가한다. 주어진 주제에 대해 자신의 의견을 말하거나, 요약하고 해결책을 제시한다. 그밖에 듣기(listening) 영역은 대화나 강의의 주된 내용을 묻는 문제, 목적을 묻는 문제, 특정한 세부 정보를 묻는 문제, 화자의 의도나 목적을 묻는 문제, 화자의 태도나 의견을 묻는 문제, 대화나 강의의 구조나 전개 방식을 묻는 문제, 내용을 연결하여 이해하는 문제, 명시적으로 언급되지 않은 정보를 추론하는 문제가 있다[28].

영어의 문제 영역을 분석하여 컴퓨터 프로그래밍 분야에서 이미 다루어진 문제 유형과 아직 다루어지지 않은 문제 유형을 식별할 필요가 있다. 만일 컴퓨터 프로그래밍 분야에서 다루어지지 않은 문제 유형이 있다면 이를 새롭게 개발할 필요가 있다. 이 내용을 Table 1과 같이 요약하였다. 영어의 읽기는 코딩에서 주어진 코드를 이해하고 분석하는 능력을 의미한다. 이는 소스 코드의 기능, 논리 또는 결과를 해석하는 작업을 포함한다. 영어의 쓰기는 코딩에서 문제를 해결하기 위해 새로운 코드를 작성하고 알고리즘을 구현하는 것을 의미한다. 이는 특정 기능을 수행하거나 주어진 문제를 해결하기 위해 코드를 작성하는 작업을 포함한다. 영어의 듣기는 각 프로그램의 목적을 파악하고

내용을 연결하여 이해하는 것이다. 영어의 말하기는 전체적인 코드 분석을 수행하여 그 의미가 무엇인지 말로 표현하는 것이다.

영어 역량을 위해 언급된 4가지 활동이 모두 다루어지고 통합되어야 하는 것처럼[29], 프로그래밍 역량을 위해서도 다양한 활동이 모두 다루어지고 통합되어야 한다. 많은 연구에서 프로그래밍 역량을 읽기(이해), 쓰기(코딩), 디버깅하기로 정의했음에도 불구하고, 기존의 프로그래밍 영역의 문제는 주로 쓰기(작성)와 코드를 따라하는데 중점을 두는 교육이 진행되고 있다. 아울러 영어와 달리, 프로그래밍에서는 읽기(이해)의 영역이 실행 결과의 예측에 초점을 맞추고 있다. 본 연구는 3장에서 Table 1에 기술된 영역들을 위주로 문제의 유형을 정의하여 객관식 유형의 문제를 생성하고자 한다.

Table 1. Problem type comparisons between TOEFL and computer programming languages

TOEFL	Programming Test	Description
reading	Code Comprehension and Analysis	<ul style="list-style-type: none"> <li>- to include the ability to understand and analyze the given code.</li> <li>- to interpret the functionality, logic, and outcome of the source code.</li> </ul>
writing	Coding	<ul style="list-style-type: none"> <li>- to write new code and implements algorithms to solve problems.</li> <li>- to write code to perform specific functions or solve given problems.</li> </ul>
listening	Code Inspection/Walk-through	<ul style="list-style-type: none"> <li>- to identify the purpose of the program.</li> <li>- to connect and understand the content, perform an overall code analysis.</li> </ul>
speaking		<ul style="list-style-type: none"> <li>- to express the meaning of the entire code verbally.</li> </ul>

### 2.2 프로그램의 난이도 측정에 관한 연구

소프트웨어의 척도(metric) 중, 정량적인 방법에 근간을 둔 사이클로매틱 수(cyclomatic number)[18]와 Halstead의 소프트웨어 복잡도[19] 계산 방식은 다음과 같다. 먼저 사이클로매틱수는 Fig. 1과 같이 프로그램에 포함된 선택문과 반복문에 의해서 생기는 제어흐름도 상의 닫힌 영역의 수를 의미한다[18]. 결국, 선택문과 반복문이 많을수록 프로그램의 복잡도는 높아진다[18].

Halstead의 소프트웨어 복잡도 계산 방식에서는 프로그램이 포함한 연산자와 피연산자의 종류 및 전체 사용 횟수를 이용하여 복잡도를 계산한다[19]. 즉, 프로그램에서 변수를 많이 사용하고 그 변수의 사용횟수가 많으면 프로그램이 복잡해진다는 것이다. 연산자의 경우도 마찬가지이다. 예를 들어, 한 프로그램에서 사용된 변수, 상수, 리터럴의 수가  $n_1$ , 연산자의 수가  $n_2$ , 변수의 총 사용횟수  $N_1$ , 연

산자의 총 사용횟수가 N2일 때, Halstead는 프로그램의 길이(length)  $N=(N1+N2)$ 로, 단어의 수  $n=(n1+n2)$ 로 정의하였다. 또한, 프로그램의 부피(volume)  $V=N*\log2n$ , 노력(effort)  $E=(V/2)*n2$ 로 정의하였다[19]. 이 두 방법을 기반으로 생성형 AI의 문제 생성을 위한 프롬프트를 생성한다면, 난이도가 서로 다른 문제들을 만들 수 있다.

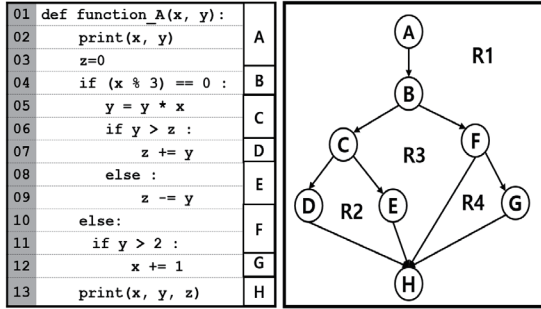


Figure 1. An example for a cyclomatic number

### 3. 문제의 설계와 구현

#### 3.1 프로그래밍 언어의 내용 요소와 난이도

고등학교에서 다루어지는 파이썬 프로그래밍 언어의 구성요소는 다음 Table 2와 같다. Table 2의 요소는 2022 개정 교육과정에서 정의된 중·고등학교의 내용 요소를 바탕으로 하였다[30]. 내용 요소로 프로그래밍 요소인 변수와 상수, 리터럴(literal)과 같은 식별자, 데이터형, 연산자, 식과 제어문, 함수를 선정하였다. 본 연구에서는 소프트웨어 척도 방법을 바탕으로 문항을 개발할 때, 난이도를 측정하기 위해 크게 정량적인 항목과 정성적인 항목으로 구분하였다.

Table 2. Programming components learned in high schools

Component	Content
Identifiers	variable, constant, literal
Data types	int, float, str, bool, list, dict, set, tuple
Operators	arithmetic, comparison, and logical operators
Expressions	operands, operators, priorities, values
Statements	sequence, selection, iteration
Functions	user defined functions

정량적인 항목의 하위 항목으로는 Table 3과 같이 첫째, 변수의 개수, 연산자의 개수, 수식의 개수, 제어문의 개수, 함수의 개수, 함수의 매개변수의 개수를 정의하였다. 제어문의 경우는 if문만 사용, if-else문만 사용, if-elif-else문만 사용, for 또는 while의 직렬 구조와 중첩 구조 등도 함께 고려하였다. 둘째, 프로그램의 길이, 함수의 길

이, 리스트/딕셔너리의 원소와 같은 데이터의 길이를 정의하였다. 셋째, 1차 리스트와 2차 리스트와 같은 차원 구조가 난이도에 영향을 미친다고 정의하였다.

정성적인 항목으로는 문제 유형 중 일부 코드 작성 유형에서 공백에 적절한 코드를 채워 넣어야 하는 경우, 공백의 문장이 문제 내에 오는 위치를 난이도에 영향을 미치는 요소로 정의하였다. 즉, 공백의 문장이 프로그램의 앞부분에 나오는지 뒷부분에 나오는지에 따라 난이도에 영향을 미칠 수 있다. 또한, 제어문의 구조나 자료구조가 난이도에 영향을 미친다. 즉, 제어문의 구조가 직렬인지 중첩인지에 따라 난이도가 달라질 수 있다. 마지막으로 함수에서 영역으로 전역변수를 사용하는 경우와 지역변수를 사용하는 경우가 프로그램의 난이도에 영향을 미친다고 정의하였다.

Table 3. Items for determining difficulty levels

Items		Description
Quantitative	Number	number of variables, operators, expressions, if, for, while statements, functions, arguments
	Length	statements, functions, list and dictionary elements
	Dimensions	dimensions of lists and dictionaries
Qualitative	position	position of the blanks to be filled in with code
	structure	nested-if, nested-for, mixed structure, composite data structure
	scope	global variable, local variable

본 연구에서는 GPT(ChatGPT 4o) 기반의 문제를 생성할 때, 자동으로 프롬프트를 생성할 수 있게 했다. 이때, Table 3에서 정의한 항목들을 이용하여 변수의 개수를 정하고, 프로그램의 길이, 함수의 개수, 제어문의 개수, 위치, 구조, 영역 등을 정의한 후, 문제를 생성하게 하였다.

#### 3.2 문제 유형

##### 3.2.1 읽기 유형

본 연구에서는 TOEFL에서 읽기 평가 문항의 유형을 반영하여 기존에 실행 결과 위주의 문제 유형을 확대하여 다음 Table 4와 같이 읽기 문제의 유형을 정의하였다.

첫 번째 유형은 특정 변수와 수식, 함수가 어떤 값을 갖는지 묻는 문제이다. 이는 전통적인 문제 유형 중 하나이다[12]. 또한, 과거에는 다루어지지 않았던 프로그래밍 언어의 영역(scope)의 개념을 프로그래밍으로 질문하는 문제를 포함하였다. 영역의 주제는 현재 시점에서 특정 변수가 어떤 영역 내에 정의된 변수인지를 이해하는 문제이다.

두 번째는 오류를 찾는 문제 유형이다. 이 문제 유형은 TOEFL의 거짓 정보(negative factual information) 문제의 유형으로부터 착안된 것이다. 프로그래밍 역량에

서는 읽기뿐만 아니라 디버깅 역량도 중요하다는 연구 결과[31-33]가 많다. 본 연구에서는 구문(syntax) 오류, 의미(semantics) 오류, 논리(logical) 오류를 포함하였다. 구문 오류는 프로그램이 올바른 구문을 따르지 않았을 때 발생한다. 의미 오류는 프로그램이 프로그래머의 의도와 다르게 동작할 때 발생한다. 즉, 코드가 문법적으로는 올바르지만, 프로그래머가 의도한 작업을 수행하지 않는 경우다. 논리 오류는 프로그램이 문법적으로 올바르고 프로그래머의 의도대로 코드가 실행되지만, 그 논리가 잘못되어 예상치 못한 결과를 생성할 때 발생한다. 의미 오류는 프로그래머가 언어의 사용법을 잘못 이해해서 발생하는 반면, 논리 오류는 문제를 해결하기 위한 접근 방식이나 알고리즘의 구현에 문제가 있을 때 발생한다.

세 번째는 전통적인 프로그램의 실행 결과를 묻는 문제 유형이다. 읽기 유형의 문제 중 논리 오류를 찾는 경우에는 프롬프트에서 생성할 프로그램의 의미를 추가로 설명하였다.

Table 4. Problem types for program reading

Types	Description
Value findings	- values of specific variables, expressions, or functions - global and local variables' values
Error findings	- the positions containing syntactic, semantic, logical errors
Program execution result findings	- the results of programs after executions

### 3.2.2 작성하기 유형

본 연구에서는 코드 작성하기 유형의 객관식 문제 유형으로 Table 5와 같이 빈칸 채우기, 코드 조각 맞추기, 단순화하기 문제를 제시하였다. 수행평가 시, 전체 프로그램을 계획하고 작성하는 과정을 학습하여 본 연구는 전체 프로그램을 모두 작성하는 것은 제외하고, 일부 프로그램을 이해하고 작성하는 문제에 초점을 두었다. 일부 프로그램을 작성하는 문제는 이미 제시된 문장들에 관한 이해가 필요하여, 읽기와 쓰기 역량을 함께 신장시킬 수 있는 장점이 있다. 또한, 코드 조각들을 제시하여 문제 해결을 위해 코드 조각의 순서를 맞추어보는 문제도 읽기와 쓰기 역량을 함께 신장시킬 수 있다. 작성하기 유형의 문제에서도 프롬프트에 문제의 의미를 설명하는 부분이 추가되었다.

마지막 유형은 TOEFL에서 복잡한 문장을 단순하게 바꾸는 단순화(simplification)를 프로그래밍에서 코드 리팩토링(refactoring)으로 정의하였다. 프로그램의 리팩토링이란 프로그램의 수행결과에 변경 없이 코드의 구조를 효율적으로 재조직하는 것을 의미한다[34]. 소프트웨어의 효율을 높이기 위해 필요한 것으로 코드 작성유형에 포함시켜 정의할 수 있다. 본 연구에서는 단순화 유형은 정의만 하였고, 난이도가 높다고 판단이 되어 실험에서는 제외하였다.

Table 5. Problem types for program writing

Types	Description
Filling in the blanks	- to guess the variable names - to guess the operators - to guess the conditions in the control statement - to guess the correct statements - to guess the values of the function parameters
Ordering	- to rearrange the code snippets in the correct order
Simplifying	- code refactoring

### 3.2.3 듣기와 말하기 유형

본 연구에서는 듣기와 말하기 유형으로 Table 6과 같다. 이 유형에는 이 프로그램이 무엇을 계산하기 위한 것인지 한마디로 말하기, 코드를 보고 발생할 수 있는 예외 상황을 말하기, 시간복잡도를 말하기 문제를 제시하였다.

Table 6. Problem types for program listening and speaking

Types	Description
Program meaning finding	- programs' purposes or meanings
Program exception finding	- programs' exceptions the programs contain.
Program complexity finding	- programs' time complexities - programs' space complexities

듣기와 말하기 유형에서는 특히 프로그램의 논리적인 측면이 강조될 수 있는 특징을 가지므로, 프롬프트에 프로그램의 의미를 설명하는 부분을 추가하였다.

## 4. 평가 문항의 적합성 검증

### 4.1 평가 문항의 개발

본 연구에서는 생성형 AI인 ChatGPT 4o[35]를 이용하여 3장에서 정의한 문제 유형별로 50개의 객관식 문항을 생성한 후, 연구 [26]의 결과를 바탕으로 20개의 문항을 Table 7과 같이 마련하였다.

Table 7. Question description

No	Description							DL
	A	B	C	D	E	F	G	
p1	2	2	0	0	0	0	×	L
p2	4	3	1	0	0	0	×	M
p3	4	3	1	1	0	1	×	M
p4	5	4	0	0	1	0	×	M
p5	5	4	0	0	0	0	×	M
p6	2	1	0	0	0	1	○	H
p7	3	2	1	1	1	1	×	M

No	Description							DL
	A	B	C	D	E	F	G	
p8	1	4	0	1	0	0	×	M
p9	4	1	0	1	0	1	×	M
p10	3	2	0	2	0	2	×	H
p11	2	3	0	1	1	0	×	M
p12	4	3	1	0	2	0	×	H
p13	3	2	1	1	0	2	×	H
p14	2	5	0	1	0	0	×	M
p15	4	3	1	0	0	1	×	M
p16	3	3	0	1	0	1	×	M
p17	2	2	0	1	0	0	×	M
p18	3	4	0	0	1	0	×	M
p19	4	5	0	2	0	0	×	H
p20	4	3	1	2	0	1	×	H

A: No. of variables, B: No. of operators, C: No. of ifs,  
 D: No. of fors or whites, E: No. of functions,  
 F: No. of lists or dictionary, G: Scope(local/global)  
 DL: Difficulty level(H: High, M: Middle, L: Low)

본 연구에서 생성된 문제 유형별 문제의 예시는 Fig. 2와 같고, 사용한 프롬프트는 다음과 같은 방식으로 정의하였다.

2. What is the result of running the following program?

```

01 a = 10
02 b = 20
03
04 result = a * b << 2
05 message="big"
06 if result > 1000:
07     message = "large"
08 elif result > 500:
09     message = "medium"
10 else:
11     message = "small"
12
13 print(result, message)
    
```

200 small  
 400 small  
 800 medium  
 1200 large  
 no answers

7. The following program finds the maximum value in a list of elements, and there is a logic error in the program. What is the line number where the error is?

```

01 def findmax(L):
02     max = 100
03     for number in L:
04         if max < number : max = number
05     return max
06 print(findmax([70, 20, 90]))
    
```

line number 01  
 line number 02  
 line number 03  
 line number 04  
 no answers

10. Explain what task the following program is designed to perform. The result of running the program is 4 3 4 5, printed vertically.

```

01 words = ["apple", "banana", "cherry", "date"]
02 for word in words:
03     v = set(word)
04     print(len(v))
    
```

This is a program that finds the length of each element in the list words.  
 This is a program that calculates the number of different alphabets each element of the list words has.  
 This is a program that calculates the number of vowels (a, e, i, o, u) that each element of the list words has.  
 This is a program that calculates the number of consonants, not groups, that each element of the list words has.  
 no answers


13. The following program is a code to create a list by collecting only positive numbers from a list and print the number of elements in the list. Write the Python code that should go in the blanks (a) and (b).

```

01 numbers = [10, -1, 12, -20, 15, -3]
02 p = []
03 for n in numbers:
04     if n <= 0 : (a) _____
05     p.append(n)
06 print (b) _____
    
```

(a) break (b) numbers  
 (a) break (b) p  
 (a) continue (b) numbers  
 (a) continue (b) p  
 No answers

17. The following code snippet is for drawing a star. Put the code snippets in the correct order, referring to the turtle's head. Indentation is not shown.



code snippet	code
①	import turtle as t
②	t.rt(144)
③	for i in range(5):
④	t.fd(100)

①→②→③→④  
 ①→②→④→③  
 ①→③→②→④  
 ①→③→④→②  
 No answers

Figure 2. Examples for evaluation problems

사용한 프롬프트는 먼저 문제의 유형(프로그램 실행 결과 찾기, 오류 찾기, 빈칸 채우기, 순서 맞추기, 프로그램 의미 찾기)을 선택한 후, 변수의 개수, 연산자의 개수, 선택문/반복문의 개수, 함수의 개수를 입력하게 한 연구[26]를 기반으로 하였다. 연구 [26]에서는 고등학교 정보 교과서 [36, 37]의 프로그래밍 예제를 GPT-3.5-turbo 모델로 파인 튜닝하여 자동 프롬프트로 사용자가 원하는 문제와 5개의 객관식 선지 및 정답을 생성한 후, 생성된 문제의 정확성을 검증하였다. 검증 결과, 프롬프트를 기반으로 문제의 정확성이 70%로 평가되어 아직은 문항을 완전히 생성형 AI에만 의존할 수 없어서 저자가 전체 문항을 분석하여 수정하여 활용하였다.

다음은 중등교사 24명을 대상으로 20문항을 해결하고 직접 난이도를 정하도록 요구하였다. 20문항에서 프로그램 실행 결과 찾기(1~5), 오류(error) 찾기(6~9), 프로그램의 의미 찾기(10~12), 빈칸 채우기(13~16, 19~20), 코드 순서 맞추기(17~18) 문제를 줬다. 본 연구에서는 프롬프트에서 주어져야 할 난이도 측정을 위한 기준을 제시하기 위해 3장 1절에서 정의한 정량적인 난이도 항목들을 적용하여 Table 7에 함께 제시하였다. 이 중, 변수와 연산자의 경우는 1~3개를 사용하면 하(low), 4~5개를 사용하면 중(middle), 그 이상은 상(high)으로 두었다. 제어문과 리스트 디셔너리, 함수의 경우는 0개를 하, 1개를 중, 2개 이상을 상으로 두었다. 영역의 사용은 1개 이상 사용시 모두 상으로 분류하였다. 만약 난이도가 상충하는 경우, 더 높은 쪽의 난이도를 최종 난이도로 채택하였다.

## 4.2 평가 결과 분석

### 4.2.1 정보교사의 난이도 평가

교사들이 문제를 해결한 후, 난이도를 측정한 결과는 Table 8과 같았다. Table 7의 문제에 관한 기술과 Table 8의 맞힌 명수와 난이도를 비교하면, 다음과 같은 특징이 있음을 알 수 있다. 첫째, 변수의 수와 연산자의 수는 적을 수록 난이도가 낮았다. 둘째, 구성요소가 많을수록 난이도가 높아짐을 알 수 있었다. 특히, 선택문보다는 반복문이 2개인 문제의 유형과 리스트가 포함된 문제가 높은 난이도를 가졌다. 셋째, 함수가 있는 문제는 대부분 교사가 ‘중’ 이상의 난이도를 표시하였다. 넷째, 본 연구에서 제시한 난이도 기준과 교사가 측정한 난이도 기준이 상이한 문제는 P6, P9, P10, P13, P20이었다.

P6의 경우, 본 연구에서는 변수의 영역이 사용되면 난이도를 상으로 하였으나, 교사들은 중으로 분류하였다. 영역에 대한 기준을 사용 여부로 정하지 않고 정략적 기준을 적용할 수 있음을 알 수 있다.

P9와 P10의 경우 본 연구에서 측정한 난이도와 서로 다를 수 있다. 이 경우, P9은 오류 찾기, P10은 프로그램의 의미를 찾는 문제 유형으로 서로 문제 유형이 다를 수 있고, 난이도가 일치하지 않는 것으로 미루어보아 문제 유형이 난이도에 영향을 미치는 것으로 나타났다. 문제 유형에 관한 분석은 다음 절에서 제시하였다.

P13과 P20은 모두 빈칸 채우기의 문제로 제어문의 수와 리스트의 개수가 각각 2 이상으로 본 연구에서는 상으로 분류하였으나, 교사는 중으로 분류하였다. 두 문제의 공통점은 전체 프로그램의 길이가 6 정도로 본 연구에서는 프로그램의 길이를 난이도에 적용하지 않았으나, 교사의 응답으로부터 난이도에 프로그램의 길이도 영향을 미침을 파악하였다. 교사가 상으로 분류한 문항의 프로그램 길이는 모두 10 이상으로 조사되었다.

Table 8. No of correct answers for 20 questions and difficulty levels

No	No of Corrects	DL1			DL2
		High	Middle	Low	
p1	16	0	0	24	L
p2	19	8	15	1	M
p3	15	6	17	2	M
p4	24	1	13	10	M
p5	13	1	12	11	M
p6	16	8	15	1	H
p7	11	2	15	7	M
p8	18	5	12	7	M
p9	13	12	11	1	M
p10	21	8	10	6	H
p11	23	3	13	8	M
p12	13	11	9	4	H
p13	16	5	15	4	H

No	No of Corrects	DL1			DL2
		High	Middle	Low	
p14	21	3	13	8	M
p15	16	11	11	2	M
p16	9	5	15	4	M
p17	19	3	12	9	M
p18	18	6	14	4	M
p19	11	17	7	0	H
p20	15	5	15	5	H

DL1: Difficulty level evaluated by teachers

DL2: Difficulty level evaluated by this research

### 4.2.2 문제 유형별 맞힌 점수 평균 차이 분석

다음은 문제 유형별 맞힌 문제에 대한 점수 평균의 차이를 살펴보기 위해, Table 9와 같이 분산분석을 실시하였다 (Fig. 3 참조). 분석 결과는  $F = 3.098$ ,  $p = 0.018$ 로 유의수준 0.05에서 통계적으로 유의미하게 문제 유형별 난이도에 차이가 드러났다. 하지만, Scheffe 기반의 사후 검정에서는 문항 간 통계적인 차이는 유의미하지 않았다.

교사들이 어려운 문제는 오류를 찾는 문제와 빈칸 채우기 문제였으며, 상대적으로 쉬운 문제는 프로그램의 의미를 찾는 문제와 코드 조각의 순서 맞추기 문제였다. 느낀 점을 묻는 주관식 문항에서 코드조각 맞추기 문제와 프로그램의 의미를 찾는 문제는 객관식 문항의 답에 대한 선택지로부터 유추가 가능했다고 응답했다.

Table 9. Result of ANOVA for different problem types

Types		Mean	SD
reading	program execution result finding (A)	7.25	2.19
	error finding (B)	6.11	1.82
writing	fill in the blanks (C)	6.04	2.75
	ordering (D)	7.71	3.29
program meaning finding (E)		7.92	1.92
total		7.01	2.54

	sum of square	degree of freedom	mean square	F	p
between groups	74.776	4	18.69	3.098	.018
within groups	693.935	115	6.03		
total	768.712	119			

(I) type	(J) type	I-J	p
error (B)	reading(A)	-1.208	.091
	writing(C)	-0.069	.922
	speaking(E)	-1.877*	.009
	order(D)	-1.667*	.020
writing (C)	reading(A)	-1.139	.111
	speaking(E)	-1.807*	.012
	error(B)	0.069	.922
	order(D)	-1.597*	.026

(I) type	(J) type	I-J	p
order (D)	reading(A)	0.458	.519
	writing(C)	1.597*	.026
	speaking(E)	-0.210	.768
	error(B)	1.667*	.020
speaking (E)	reading(A)	0.668	.348
	writing(C)	1.807*	.012
	error(B)	1.877*	.009
	order(D)	0.210	.768

\*significantly statistically different at  $\alpha = 0.05$

(A: program execution result finding problems, B: error finding problems, C: filling in the blanks problems, D: ordering problems, E: program meaning finding problems.)

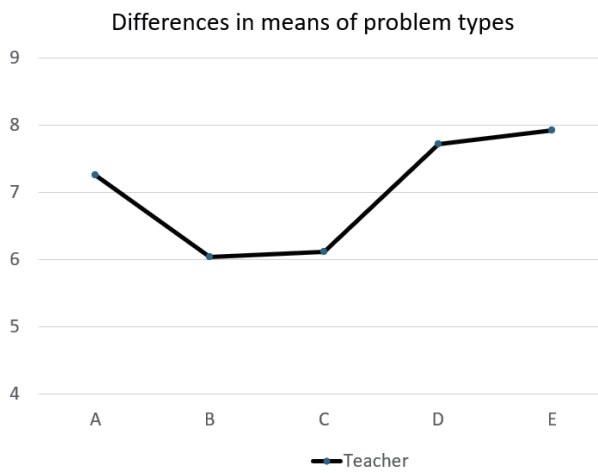


Figure 3. Differences in means of problem types for teachers

반면, 프로그램의 모든 구문을 다 알지 못하는 상황에서 오류를 찾는 문제는 상대적으로 어려웠으며, 그동안 경험하지 못하였음을 알 수 있었다.

문제를 해결한 한 교사는 다음과 같은 주관적인 의견을 제시하였다. ‘학생들이 객관식 문제를 해결하는 과정에서 답의 대안들이 답을 추론하는데 많이 작용을 하여, 프로그램의 의미를 파악하는 문제가 쉬울 수 있다. 최근 지필평가에서 객관식 문제와 더불어 주관식 문제가 필수화되었다. 이에 따라서 문제 유형별로 결과를 맞히거나 오류를 찾는 것은 객관식 문제 유형이거나 주관식 문제 유형일 때, 큰 차이가 없을 수 있지만, 프로그램의 의미를 맞히는 문제는 주관식 문제가 더욱 적절하다고 생각한다.’라고 응답하였다.

결국, 문제를 제시할 때 정확한 평가를 위해서는 객관식 문항에 적합한 문제 유형과 주관식 문항에 적합한 문제 유형이 각각 있을 수 있음을 알 수 있다.

## 5. 결론

본 연구에서는 고등학생들이 배우는 파이썬 언어를 중심으로 다양한 유형의 프로그래밍 문제 유형을 정의하고 난이도를 측정할 방법을 제안하였다. 본 연구는 프로그래밍 언어 능력 평가를 일반 외국어와 유사한 시각에서 접근

하고, 아직 다루어지지 않은 문제 유형을 탐색하고 개발하였다.

본 연구에서는 TOEFL의 문제 유형 중에서 제시되지 않거나 틀린 정보를 고르는 거짓 정보 문제 유형에 주목하고, 여기에 듣기와 말하기 유형의 문제를 추가하였다. 또한, 문항에 대해 교사에게 객관식 문제를 제시하였다.

난이도의 경우, 본 연구에서는 정량적인 항목과 정성적인 항목을 제시하였다. 하지만, 모든 난이도 측정 항목을 적용하지는 못하였으며, 일부의 난이도 측정 항목만을 사용하여 교사의 응답과 비교하였다. 응답 중, 난이도가 일치하지 않은 문항의 특성을 분석해 본 결과, 본 연구에서 제시한 항목 이외에 프로그램의 전체 길이와 문제 유형이 함께 고려되어야 함을 알 수 있다. 추후 본 연구에서 다루지 못한 항목을 추가하여 정량적이고 정성적인 항목을 포함하는 좀 더 체계적인 난이도 척도의 개발이 필요하다.

또한, 교사와 학생이 모두 오류를 찾는 문제를 코드 작성 문제보다 더 어렵게 느꼈다. 이는 아직도 프로그래밍 교육에서 학생들이 디버깅에 관한 훈련이 부족함을 의미하며, 교사들도 실제 교육 시간을 디버깅 과정에 많이 할애하고 있지 않다고 응답하였다. 교육 현장에서 좀 더 많은 디버깅에 관한 교육이 필요함을 알 수 있다.

마지막으로는 사이클로매틱 수와 Halstead의 프로그램 척도를 기준으로 분석한 결과, 정량적인 프로그래밍 내용 요소의 개수가 많을수록 교사들이 난이도를 상으로 매김 한 것으로 드러났다. 교사의 반응으로부터 단지 정량적인 요소의 프로그래밍 문제 보다 프로그램의 의미가 들어간 정성적인 요소를 포함시킬 필요가 있음을 알 수 있다.

문제 해결 후, 교사들의 의견을 묻는 문항에서 생성형 AI를 이용한 문제생성 방식에 대해 처음에는 우려의 목소리가 있었으나, 교사 자신이 생각하지 못한 창의적인 문제를 발견하면서 문제 출제 방식을 좀 더 확장할 수 있었다는 긍정의 응답이 있었다. 반면, 여러 내용 요소를 조합하여 문제를 제시하다 보니, 평가 요소가 불명확하다는 부정적인 응답도 있었다. 특히, 내용 요소의 개수를 통해 자동 프롬프트를 생성하여 문제를 만드는 방식은 편리할 수 있으나, 프로그램의 의미를 살리는 데는 한계가 있어서 생성형 AI 활용에 관한 추가적인 연구가 필요하다.

다양한 문제 유형의 필요성에 대해서는 공감하였고, 지필 평가가 객관식과 주관식으로 이루어질 경우, 각 유형에 적합한 문제 형식을 잘 판단하는 것이 중요하게 나타났다. 또한, 복잡도와 관련하여 프로그래밍 내용 요소의 개수와는 별개로 함수에 관한 문제는 무조건 ‘중’ 이상의 난이도를 가지게 된다는 의견이 있어, 정성적인 척도의 역할을 할 수 있다. 본 연구에서는 문제의 유형은 제안했으나, 모든 유형의 문제를 다 제시하지는 않았다. 적절한 난이도를 갖는 다양한 유형의 문제 유형을 갖는 문항을 생성하여 문항의 타당도와 신뢰도 등 적합성을 분석하는 연구가 필요하다. 또한 본 연구는 고등학생의 프로그램 교육과 관련되었으나 연구 결과를 소수 교사 대상으로 하였다라는 점에서 한



계점이 있다. 향후 고등학생으로 확장하여 본 연구 결과를 일반화할 필요가 있다.

영어 공부에서 읽기, 쓰기, 듣기, 말하기를 구분하는 것은 흔하지만, 실제 언어 사용에서는 이런 구분이 명확하지 않다. 이렇게 서로 다른 영역을 넘나드는 것이 언어 사용의 역동성을 더해준다[25]. 결국 컴퓨터 프로그래밍 교육에서도 코드의 작성과 코드의 이해, 코드의 디버깅이 함께 진행될 수 있다면, 학생들의 프로그래밍 역량을 높이는 데 도움을 줄 수 있다.

## 참고문헌

- [1] Hong, H. J. (2023). *Development of an Artificial Intelligence-Based Convergence Transportation Technology Problems Solving Program in Middle School Technology Subjects for Enhancing Computational Thinking Ability*. [Master's Thesis]. Graduate School of Education, Jeju National University.
- [2] Kim, M. S. (2021). *The Effect of PBL-based EPL Programming Learning on Improving Problem Solving Ability and Logical Thinking Ability*. [Master's Thesis]. Graduate School of Education, Jeonnam National University.
- [3] Kim, S. H. (2015). Analysis of Non-computer Major's Difficulties in Computational Thinking Education. *Journal of Korean Association of Computer Education*, 18(3), 49-57.
- [4] Oh, K., & Ahn, S. (2015). A Study on the Relationship between Difficulty in Learning to Program and Computational Thinking. *Journal of Korean Association of Computer Education*, 18(5), 55-62.
- [5] Kim, J., & Sohn, E. (2021). Difficulty Analysis of an Introductory Computer Programming Course for Non-major Students. *Journal of Creative Information Culture*, 7(2), 69-77.
- [6] Lee, T. (2023). *A Study on the Python Education Program Using Artificial Intelligence Pair Programming*. Graduate School of Education. [Master's thesis]. Sungkyunkwan University.
- [7] Park, J. (2023). *Development of Classification Project Classes using Artificial Intelligence Platform with Python Error Data for High School Students*. [Master's thesis]. Graduate School of Education, Korea National University of Education.
- [8] Moon, W. S. (2008). Influential Error Factors of Robot Programming Learning on the Problem Solving Skill. *Journal of Korean Association of Information Education*, 12(2), 195-202.
- [9] Kim, Y. J., & Hong, H. G. (2019). A Comparative Analysis of the Evaluation System of High Schools in Korea and Finland. *Journal of Curriculum and Evaluation*, 22(1), 77-100. DOI: G704-000854.2008.12. 2.008.10.22799/jce.2019.22.1.004
- [10] Kim, H. W., Yun, H. J., & Kim, K. H. (2024). A Study on the Intelligent Online Judging System Using User-based Collaborative Filtering. *Journal of The Korea Society of Computer and Information*, 29(1), 273-285. <https://doi.org/10.9708/jksci.2024.29.01.273>
- [11] Han, S., Liu, X., & Woo, G. (2022). Evaluation Methods and Classification of Online Judge Systems Based on Education Level and Purpose. *KIISE Transactions on Computing Practices*, 28(10), 487-492. <https://doi.org/10.5626/KTCP.2022.28.10.487>
- [12] Park, C. J., & Hyun, J. S. (2024). Analysis of Conjoint-Based Teaching Method Preferences for Perspective of Debugging Activities in Elementary and Secondary Programming Education. *The Journal of the Korea Contents Association*, 24(2), 476-486. <https://doi.org/10.5392/JKCA.2024.24.02.476>
- [13] Park, B., & No, G. (2024). The Effects of Integrating English and Educational Programming Languages on English Learning. *Studies in Foreign Language Education*, 38(3), 209-232. <http://dx.doi.org/10.16933/sfle.2024.38.3.209>
- [14] Hyun, J. S., & Park, C. J. (2023). Construction of Test Questions in Computer Programming Education from the Perspective of Error Handling. *Proceedings of 2023 International Conference on Convergence Content (ICCC 2023)*. 21(2), 163-164.
- [15] Seol, H. S. (2023). snowIRT: Development of an Open Source Statistical Program for Distractor Analysis. *Korean Education Inquiry*, 41(4), 103-119.
- [16] Kim, J., & Kim, Y. (2021). Development and Validation of Creative and Critical Thinking Competency Diagnostic Scale Using Item Response Theory. *Korean Journal of General Education*, 16(5), 317-331. <https://doi.org/10.46392/kjge.2022.16.5.317>
- [17] Kim, E. (2022). A Study on the Difficulty Adjustment of Programming Language Multiple-Choice Problems Using Machine Learning. *Journal of the Korea Industrial Information Systems Research*, 27(2), 11-24. <http://dx.doi.org/10.9723/jksis.2022.27.2.011>
- [15] Elshoff, J. L., & Marcotty, M. (1978). On the Use of the Cyclomatic Number to Measure Program Complexity. *ACM SIGPLAN Notices*, 13(12), 29-40.
- [16] Hariprasad, T., Vidhyagaran, G., Seenu, K., & Thirumalai, C. (2017). Software Complexity Analysis Using Halstead Metrics. *2017 IEEE International Conference on Trends in Electronics and Informatics (ICEI)*, pp. 1109-1113.
- [17] Jeong, S. Go, H. N., & Lee, Y. (2024). Development and Application of Non-linear Search Questions Using an Online Judge System. *Journal of Korean Association of Computer Education*, 27(2), 1-11. <http://dx.doi.org/10.32431/kace.2024.27.2.001>
- [18] Go, H. N., Jeon, J. H., & Lee, Y. (2023). A Study on the Development of Problem Bank for Programming-Math Convergence Education in Programming Automatic Assessment System. *Journal of Korean Association of Information Education*, 27(2), 141-152.

- [19] Kim, S. S., Oh, S. H., and Jeong, S. S. (2018). Development and Application of Problem Bank of Problem Solving Programming Using Online Judge System in Data Structure Education. *Journal of Korean Association of Computer Education*, 21(4), 11-20. <http://dx.doi.org/10.32431/kace.2018.21.4.002>
- [20] Feng, Y. H., & Jung, J. H.. (2024). A Feasibility Study on Relief Creation Methods Based on Generative AI. *Journal of Basic Design & Art*, 25(2), 123-140.
- [21] Kang, S. C., & Heo, H. (2023). Development and Application of Generative AI-based Instructional Design Platform. *Journal of Korean Association of Computer Education*, 26(6), 143-153. <http://dx.doi.org/10.32431/kace.2023.26.6.012>
- [22] Han, O. (2023). A Study on Components for Designing Personalized Education Systems Based on Generative AI. *Journal of Korean Association of Computer Education*, 26(6), 127-141. <http://dx.doi.org/10.32431/kace.2023.26.6.011>
- [26] Lee, Y., Kim, J., & Park, C. J. (2024). Development of a GPT-based Multiple-choice Question Generation Program for the Programming Area in High School Informatics Subject. *Journal of The Korean Association of information Education*, 28(4), 473-484.
- [27] cbtKorea. (2024, July 2). *What is TOEFL?* [https://www.cbtKorea.com/info/toefl\\_info.html](https://www.cbtKorea.com/info/toefl_info.html)
- [28] ETS. (2024, July 2). *TOEFL iBT*. <https://www.kr.ets.org/toefl/test-takers/ibt/about/content.html>
- [29] Practicus Publishing. (2024, July 2). *Let's not distinguish between reading, writing, listening, and speaking*. <https://blog.naver.com/knowhereman/40118518049>
- [30] Ministry of Education (2022). *2022 Revised National Education Curriculum*. Ministry of Education Notification No. 2022-33 [Appendix 10]
- [31] Lee, C. H. (2021). Development and Application of Customized Modular EPL Education Programs According to EPL Programming Difficulties and Error Types. *Journal of Education*, 41(3), 83-103. The Institute for Education and Research Gyeongin National University of Education. <http://dx.doi.org/10.25020/je.2021.41.3.83>
- [32] Lyu, K., & Kim, S. (2020). Development of Debugging Tasks and Tool for Process-Centered Assessment on Software Education. *Journal of Korean Association of Computer Education*, 23(4), 61-68. <http://dx.doi.org/10.32431/kace.2020.23.4.006>
- [33] Park, C. J., & Hyun, J. S. (2023). Development of Error-Solving Programming Education Method Using ChatGPT Based on Chaos Engineering Process. *Journal of Korean Association of Computer Education*, 26(6), 29-40. <http://dx.doi.org/10.32431/kace.2023.26.6.003>
- [34] Kim, D., Jung, Y., & Hong, J. E. (2016). Analysis of Refactoring Techniques and Tools for Source Code Quality Improvement. *Journal of Convergence for Information Technology*, 6(4), 137-150. <http://dx.doi.org/10.22156/CS4SMB.2016.6.4.137>
- [35] ChatGPT 4o. <https://chatgpt.com/>
- [36] Lee, Y., Yoo, H., Choi, J., Choi, W., and Park, J. (2018). *High School Informatics Textbook according to the 2015 Revised Curriculum*. Kyohaksa.
- [37] Kim, Y., Lim, J., Kim, M., Park, J., Jung, Y., and Lee, S. (2018). *High School Informatics Textbook according to the 2015 Revised Curriculum*. Keumsung Publishing Co.



박찬정

- 1988년 서강대학교 전자계산학과(공학사)
- 1990년 KAIST 전산학과(공학석사)
- 1998년 서강대학교 전자계산학과(공학박사)
- 1990~1994 한국통신 소프트웨어연구소 전임연구원
- 1998년 ~ 1999년 9월 한국통신 멀티미디어연구소 전임연구원
- 1999년 9월 ~ 현재 제주대학교 사범대학 컴퓨터교육과 교수
- 2013. 4 ~ 2015. 2 제주대학교 교육과학연구소장
- 2020. 3 ~ 2022. 2 한국컴퓨터교육학회 수석부회장
- ⊕ 관심분야 : 에듀테크, 프로그래밍 교육, 데이터 마이닝, 빅데이터 분석
- ✉ [cjpark@jejunu.ac.kr](mailto:cjpark@jejunu.ac.kr)



현정석

- 1991 서강대학교 경영학과(경영학사)
- 1993 서강대학교 대학원 경영학과(경영학석사)
- 1998 서강대학교 대학원 경영학과(경영학박사)
- 2002 ~ 현재 제주대학교 경영정보학과 교수
- 2008 ~ 제주대학교 대학을 빛낸 교수상 수상
- 2012 ~ 특허청장상 수상
- 2016 ~ 제주대학교 강의평가 최우수 교수상 수상
- ⊕ 관심분야 : 마케팅, 행동의사결정론, 트리즈, 창의성 교육, 영재교육, 인공지능
- ✉ [jshyun@jejunu.ac.kr](mailto:jshyun@jejunu.ac.kr)