ISSN 2383-6318(Print) / ISSN 2383-6326(Online)
KIISE Transactions on Computing Practices, Vol. 24, No. 12, pp. 649-661, 2018. 12
https://doi.org/10.5626/KTCP.2018.24.12.649

스몰베이직 기반 교육용 코딩 환경을 위한 오픈소스 소프트웨어 개발

(A Development of Open-Source Software for Educational Coding Environments Using Small Basic)

최 광 훈 † 김 가 영 †† 창 병 모 †††
(Kwanghoon Choi) (Gayoung Kim) (Byeong-Mo Chang)

요 약 본 연구에서 오픈소스 소프트웨어 기반 교육용 스몰베이직 코딩 환경을 개발하였다. 스몰베이직은 코딩 입문자를 위한 간단하고 배우기 쉬운 텍스트 기반 프로그래밍 언어이다. 하지만 기존의 마이크로소프트 스몰베이직 환경은 윈도우 운영체제에서만 사용해야 하고 소스 코드를 공개하지 않아 새로운 기능과 라이브러리를 추가하기 어려운 단점이 있다. 이 연구에서 윈도우, 리눅스, 맥에서 모두 사용할 수 있도록 자바로 새로운 스몰베이직 코딩 환경 마이스몰베이직을 개발하였다. 오픈소스 프로젝트이기 때문에 누구나 개발에 참여할 수 있는 장점이 있다. 이 개발을 통해 이전에 문서화되지 않았던 스몰베이직 언어의 파서 명세와 동적타입 변환에 대한 명세를 처음으로 작성하였다. 마지막으로 마이스몰베이직을 실제 코딩교육에 적용한 사례를 소개한다.

키워드: 스몰베이직, 오픈소스 소프트웨어, 프로그래밍언어, 코딩 교육

Abstract This paper proposes development of an open source Small Basic coding environment. Small Basic is a simple text-based programming language for novices, that is easy to learn and user-friendly. The existing coding environment known Microsoft Small Basic can operate only on Windows, and no one can freely contribute new functions and libraries to it because of its closed policy on source code. In this paper, we develop a new Small Basic coding environment named MySmallBasic written in Java to operate in Windows, Linux, and Mac platforms. It is an open-source project, and so it has an advantage in that everyone can participate in the project. As a by-product of this study, this paper provides formal specifications on the parser and on dynamic typing semantics for Small Basic programming language, that has never been documented. Last, this paper reports an experience of using MySmallBasic in a coding education lecture.

Keywords: small basic, open-source software, programming languages, coding education

· 이 논문은 전남대학교 학술연구비(과제번호: 2016-2823) 지원에 의하여 연구되었음

† 정 회 원 : 전남대학교 전자컴퓨터공학부 교수
kwanghoon.choi@jnu.ac.kr
†† 학생회원 : 전남대학교 전자컴퓨터공학부
kirayu15@gmail.com
††† 종신회원 : 숙명여자대학교 소프트웨어학부 교수
(Sookmyung Women's Univ.)
byeongmo@gmail.com
(Corresponding author임)

논문접수 : 2018년 8월 13일
(Received 13 August 2018)
논문수정 : 2018년 9월 15일
(Revised 15 September 2018)
심사완료 : 2018년 9월 19일
(Accepted 19 September 2018)

정보과학회 컴퓨팅의 실제 논문지 제24권 제12호(2018. 12)

1. 서 론

사회 전반에 컴퓨터의 사용이 일반화되어 이 컴퓨터를 운영하는 소프트웨어에 대해 잘 이해해야하고 더 나아가서 새로운 소프트웨어를 구성하는 능력이 필요하게 되었다. 이러한 추세에 맞추어 초중고교에서 일정시간의 코딩 교육을 의무화하고 있고 대학에서도 컴퓨터 이외의 전공자도 코딩 교육에 적극 참여하는 것이 세계적인 추세이다.

마이크로소프트의 비제에 라지(Vijaye Raji)가 개발한 스몰베이직(Small Basic)[1]은 처음 코딩을 배우는 초보자를 위한 프로그래밍 언어와 윈도우 기반 코딩 환경이다[1,2]. 코딩을 배우기 쉽게 최소한의 언어 요소만 도입하여 스몰베이직 프로그래밍언어를 설계하였고, 바로 사용할 수 있도록 코딩 환경의 메뉴가 매우 단순하게 구성되어 있어 코딩을 전혀 모르는 사람이 처음 배우기에 매우 적합하다. 이해하기 쉬운 라이브러리를 활용하여 텍스트 및 그래픽스 프로그램을 간단하게 작성할 수 있으며, 커뮤니티를 통해 배울 수 있도록 소스 프로그램을 공유할 수 있는 일종의 앱스토어를 제공한다.

스몰베이직 이외에도 다른 여러 교육용 코딩 환경이 있다. 스크래치[2]와 앱 인벤터[3]는 그래픽 환경에서 블록 그림으로 표현된 프로그램 조각을 쌓아 프로그램을 구성하면서 코딩을 경험하게 하는 도구이다. 전 세계에서 코딩 교육에 많이 활용하고 있다. 특히 초등학교에서뿐만 아니라 처음 코딩에 입문하는 어른에게도 유용한 교육용 코딩 환경이다. 파이썬[4]은 풍부한 라이브러리를 기반으로 다양한 프로그램을 작성할 수 있어 교육용 코딩 환경으로 인기가 높다. 특히 데이터 사이언스와 머신러닝 분야의 커뮤니티에서 개발한 방대한 파이썬 라이브러리를 활용하는 응용 분야를 염두에 두고 코딩 교육을 실시한다면 파이썬을 선택하는 것이 좋다.

스몰베이직 언어와 다른 교육용 코딩 환경에서 제공하는 언어를 다음과 같이 비교할 수 있다. 스크래치와 앱 인벤터와 같은 교육용 코딩 환경을 통해서도 코딩의 기초 개념을 배울 수 있지만 이러한 환경에서는 그림 기반 프로그램을 작성하며 코딩을 배운다. 하지만 대부분의 다른 프로그래밍언어는 텍스트 기반이기 때문에 실제 코딩 경험과 차이가 있다.[5] 반면에 텍스트 기반 프로그래밍언어인 파이썬은 C/C++/Java와 같은 객체지향 프로그래밍언어이고 너무 많은 기능적 특징을 보유하고 있어 초보자가 처음 코딩 개념을 배우기에 오히려 장벽이 될 수 있다. 예를 들어, 클래스와 객체는 기초 코딩 교육에서 최우선적으로 다루어야할 개념은 아니다. 이와 달리, 스몰베이직은 대부분의 프로그래밍언어와 동일하게 텍스트 기반이고 최소한의 프로그래밍언어 개념만 도입하여 설계되었기 때문에 처음 코딩을 배울 때 적합한 장점을 가지고 있다.

이러한 장점에도 불구하고 스몰베이직을 활용하여 코딩 교육을 진행하는 경우 세 가지 문제점이 있다. 첫째, 마이크로소프트에서 스몰베이직 환경을 닷넷프레임워크에 의존적으로 개발하였기 때문에 리눅스, 맥 운영체제, 웹, 안드로이드 스마트폰에서 실행할 수 없다. 둘째, 현재 스몰베이직 표준 라이브러리에 포함되어 있지 않은 로봇 제어, 소셜네트워크 연결 등 코딩 교육의 흥미를 높이기 위해 새로운 라이브러리를 추가하기를 원하더라도 개발 가이드 문서가 부족하고 내부 구조에 대한 정보가 공개되어 있지 않아 어렵다. 셋째, 마이크로소프트 스몰베이직 환경은 실행 바이너리 파일만 공개되어있고 소스 파일에 접근할 수 없기 때문에 새로운 기능을 추가할 수 없다. 예를 들어 마이크로소프트 스몰베이직은 디버깅 기능을 제공하지 않고 있으며, 외부에서 이 기능을 개발하더라도 확장이 불가능하다.

이 논문의 연구 목표는 스몰베이직 프로그래밍언어의 코딩 교육 관점의 장점을 유지하면서 앞에서 서술한 마이크로소프트 스몰베이직의 문제점들을 개선한 새로운 스몰베이직 기반 코딩 교육 플랫폼을 개발하는 것이다. 이 목표를 달성하기 위해, 오픈소스 소프트웨어 프로젝트 기반으로 특정 운영체제에 종속되지 않는 스몰베이직 코딩 환경을 개발하고, 누구나 자유롭게 커뮤니티에 참여하여 새로운 라이브러리나 기능을 기여할 수 있도록 한다. 기본적으로 마이크로소프트 스몰베이직에서 작성한 프로그램을 동일하게 동작하도록 호환성을 지원하는 해석기(Interpreter)와 표준 라이브러리를 개발하였다. Java기반으로 개발하여 윈도우뿐만 아니라 맥 운영체제와 리눅스에서 동일하게 동작하여 운영체제 종속성을 탈피하였다. 기존 스몰베이직 환경에서 제공하지 않았던 햄스터 로봇 라이브러리, Facebook/Twitter 라이브러리, SQLite 데이터베이스 라이브러리, Weka[3][6]기반 학습 라이브러리를 추가 개발하여 코딩 교육의 흥미를 유도할 수 있었다. 그리고 스몰베이직 프로그램 디버거를 개발하여 초보자가 실행 과정을 한 문장씩 따라가며 제어 흐름을 살피고 변수의 값이 변화되는 과정을 쉽게 확인할 수 있었다.

1) https://channel9.msdn.com/blogs/charles/expert-to-expert-the-basics-of-smallbasic
2) https://scratch.mit.edu
3) https://appinventor.mit.edu
4) https://www.python.org
5) An Interview with Prof. Won Kim, Gachon University,https://www.zdnet.co.kr/news/news_view.asp?artice_id=20180115104723
6) http://www.cs.waikato.ac.nz/ml/weka

오픈소스 소프트웨어 기반 스몰베이직 코딩 환경을 개발하기 위해 극복했던 기술적 문제점은 다음과 같다. 스몰베이직 언어에 대한 구문과 의미를 정의하는 공식 문서가 없다. 그러나 파서와 해석기를 개발할 때 이러한 공식적인 정의가 필요하다. 해결책으로, 기존 마이크로소프트 스몰베이직 코딩 환경과 기존에 작성된 스몰베이직 프로그램들을 모아 역 공학 분석하여 파서와 해석기의 명세를 작성하였다. 이렇게 분석한 내용을 이 논문에서 요약 정리한다.

이 연구에서 기여한 점은 다음과 같다. 첫째, 오픈소스 소프트웨어로 교육용 코딩 환경 마이스몰베이직[4]을 개발하였다. 둘째, 기존 마이크로소프트 코딩 환경에서 개발한 스몰베이직 프로그램의 호환성을 보장하고 10가지 확장 라이브러리와 소스 수준의 디버거를 개발하였다. 셋째, 스몰베이직 프로그래밍언어의 상세 명세, 특히 파서와 동적 타이핑에 관하여 처음으로 문서화하였다.

본 논문의 구성은 다음과 같다. 2장 관련연구에서 마이크로소프트 스몰베이직을 소개하고 기존 교육용 프로그래밍언어와 비교한다. 3장에서 오픈소스 소프트웨어 프로젝트 마이스몰베이직에 대하여 상세히 설명하고, 4장에서 이 프로젝트를 코딩 교육에 활용한 사례를 설명한다. 5장에서 결론 및 향후 연구를 기술한다.

2. 관련 연구

2.1 마이크로소프트 스몰베이직

마이크로소프트의 비제에 라지(Vijaye Raji)가 개발한 스몰베이직(Small Basic)[1]은 처음 코딩을 접하는 사람을 위한 교육용 프로그래밍언어이고, 현재 윈도우 기반 환경에서 코딩을 할 수 있다. 프로그래밍언어가 간단하고, 코딩 환경도 매우 단순하여 적응하기 쉽고, 다양한 라이브러리를 활용하여 프로그램을 쉽게 작성할 수 있다. 예를 들어, 스몰베이직을 이용해 플리커 웹사이트에서 지정한 주제의 사진들을 찾아 슬라이드 쇼하는 프로그램을 그림 1과 같이 간단하게 작성할 수 있다.

스몰베이직 프로그래밍언어의 특징은 다음과 같다.

- 스몰베이직의 키워드는 전체 16개, If, Then, ElseIf, Else, EndIf, While, EndWhile, For, To, Step, EndFor, Goto, Sub, EndSub, And, Or이다.
- 특별한 선언 없이 변수를 사용하고, 스코프(scope) 개념 없이 모든 변수를 전역 변수로 취급한다. 변수의 초기값은 빈 문자열 “”이다.
- 숫자, 문자열, 부울값, 배열을 지원한다. 모든 값은 문자열로 변환할 수 있고, 이러한 문자열을 다시 숫자, 부울값, 배열로 되돌릴 수 있다. 3장에서 자세히 설명 한다.
- 분기문 Goto와 서브루틴, 조건문 If, 반복문 For와 While을 지원한다. 서브루틴은 인자를 전달하고 값을 반환하는 방법을 제공하지 않는다. 문장으로서 서브루틴을 호출만을 허용하고, 식으로서 사용할 수 없다.
- 라이브러리에서 함수와 변수를 제공한다. 서브루틴과 달리 함수는 인자를 전달하고 값을 반환하는 방법을 제공하여 식으로써 함수 호출을 허용한다. 그리고 라이브러리에서 정의한 변수를 읽고 쓸 수 있다.

```
For i = 1 To 20
  pic = Flickr.GetRandomPicture("Korea")
  GraphicsWindow.DrawResizedImage(pic, 0,0, 640,480)
EndFor
```

그림 1 스몰베이직 프로그램 예제: 슬라이드 쇼
Fig. 1 A Small Basic Program for Slide Show

스몰베이직은 표 1에 나열한 20개의 표준 라이브러리를 제공한다. 텍스트와 그래픽 입출력을 비롯하여 코딩 교육의 흥미를 높일 수 있는 다양한 라이브러리를 갖추고 있다.

스몰베이직 프로그래밍언어로 코딩할 수 있는 환경은 그림 2의 마이크로소프트에서 제공하는 코딩 환경이 유일했었다. 이 코딩 환경은 파일 열기와 저장, 편집 복사와 붙여넣기, 프로그램 실행과 같이 반드시 필요하고 이해하기 쉬운 메뉴들만 포함되어 있다[7),8)]. 따라서 코딩

표 1 스몰베이직 표준 라이브러리
Table 1 The Standard Library in Small Basic

Library	Description
Array	Array functions
Clock	System clock functions
Controls	Button, TextField, MultiLine TextField
Desktop	Desktop wallpaper
Dictionary	Online dictionary
Flickr	Access to Flickr photo service
File	File and directory management
GraphicsWindow	Graphics functions and variables
ImageList	Image management
Math	Math functions
Mouse	Mouse cursor and button properties
Network	File download
Program	Program execution controls
Shapes	Movable graphics figures
Sound	Sound play, stop, pause functions
Stack	Stack functions
TextWindow	Text-console based input/output
Text	Text manipulation
Timer	Timer functions
Turtle	Turtle graphics

7) https://www.salon.com/2006/09/14/basic_2/
8) https://blogs.msdn.microsoft.com/smallbasic/2008/10/23/hello-world/

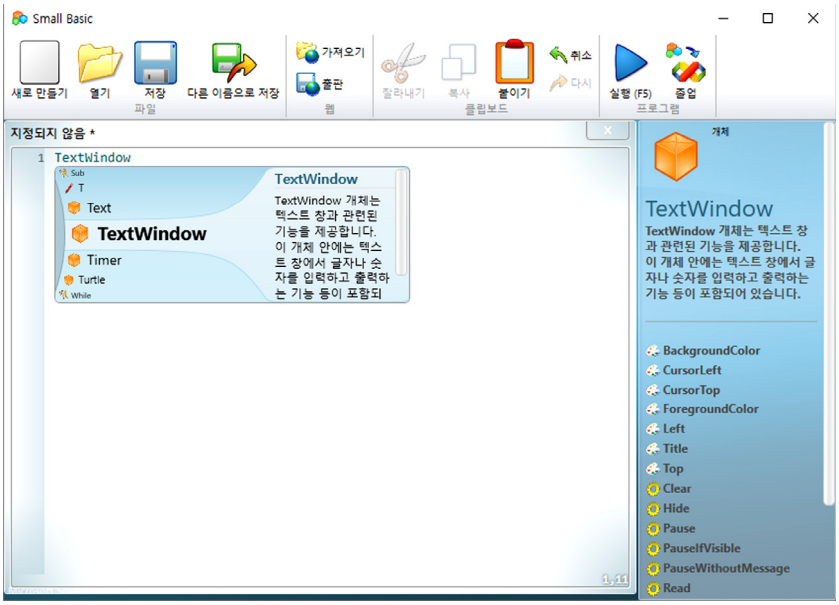

그림 2 마이크로소프트 스몰베이직 코딩 환경

Fig. 2 Microsoft SmallBasic Environment

입문자가 쉽게 사용할 수 있다. 그리고 편집 창에서 자동 완성 기능을 제공하여 라이브러리 함수와 변수의 이름을 기억하지 못하더라도 앞부분만 일부 작성하면 일치하는 함수와 변수 목록을 보여준다. 마지막으로, 스몰베이직 앱스토어에서 소스 코드를 가져오거나 자신이 작성한 소스 코드를 출판하는 메뉴를 제공한다. 스몰베이직 코딩 커뮤니티를 통해 서로 배울 수 있다.

2.2 코딩 교육을 위한 다른 프로그래밍언어와 비교

코딩 교육용 프로그래밍언어와 환경은 크게 두 가지로 구분할 수 있다. 첫째, 그림 기반 코딩 환경이 있다. 그래픽 환경에서 블록 그림으로 표현된 프로그램 조각을 조합하여 프로그램을 구성하는 방식으로, 스크래치와 앱 인벤터가 대표적인 사례이다. 특별한 사전 지식을 요구하지 않고 흥미를 높일 수 있는 요소로 구성되어 초등학교 학생 수준을 대상으로 하는 코딩 교육에 활발히 사용되고 있다. 둘째, 텍스트 기반 코딩 환경이 있다. 특히 풍부한 라이브러리를 기반으로 다양한 프로그램을 작성할 수 있는 파이썬은 교육용 코딩 언어로 많이 활용되고 있다. 데이터 사이언스와 머신 러닝 분야의 커뮤니티에서 개발한 방대한 파이썬 라이브러리를 갖추고 있기 때문에 이 분야를 목표로 한다면 교육용 코딩 언어로 파이썬을 선택할 수 있을 것이다.

표 2에서 앞서 설명한 두 가지 코딩 교육용 프로그래밍언어와 스몰베이직을 비교한 내용을 정리하였다.

스몰베이직 언어를 다른 코딩 교육용 프로그래밍언어 및 환경과 비교하면, 그림 기반 코딩 환경에 비해 텍스트 기반 프로그래밍 언어이며 다른 텍스트 기반 프로그래밍 언어에 비해 최소의 언어 특징으로 간단하게 설계되어 초보자가 처음 코딩 개념을 배우기에 적합한 장점이 있다. 그러나 현재 유일한 코딩 환경인 마이크로소프트 스몰베이직은 윈도우에서만 실행 가능하고 소스 코드를 비롯한 내부 구조를 명시적으로 공개하지 않아 새로운 라이브러리를 만들거나 디버거와 같은 새로운 기능을 개발할 때 특정 회사 개발 그룹에 의존적인 단점이 있다.

표 2 코딩 교육용 프로그래밍 언어 비교

Table 2 A Comparison for Educational Programming Languages

	Simplicity	Usability	Library
Scratch, App Inventor	Simple (Codeless)	Web, PC, Android	Limited
Microsoft Small Basic	Simple	Windows	Limited
Python, C/C++/Java	Complex	OS-neutral	Rich

대부분의 프로그래밍언어들의 공통 속성인 텍스트기반 언어를 선택한다면 스몰베이직의 장점이 분명하다. 파이썬을 비롯한 C/C++/Java는 클래스, 객체, 포인터 등과 같은 초보자의 코딩 교육에 필요한 것 이상의 언어적 특징을 보유하고 있어 배움에 오히려 장벽이 될 수 있다. 반면에 스몰베이직은 최소의 언어 특징만을 갖

추도록 설계되어 쉽게 코딩을 배울 수 있다. 스몰베이직을 코딩 교육용 환경으로 선택하는데 장애가 되는 요소는 표 2에서 비교한 것과 같이 윈도우 운영체제에서만 코딩 교육을 진행할 수 있다는 점과 다양한 라이브러리와 코딩 환경에 필요한 새로운 기능을 추가하는 것이 불가능하다.

이 논문의 연구 목표는 마이크로소프트 스몰베이직에서 정의한 프로그래밍언어의 구문과 의미를 호환되도록 유지하여 배우기 쉬운 언어의 장점을 유지하면서 오픈소스 소프트웨어 프로젝트 커뮤니티의 기여를 통해 앞서 언급한 두 가지 단점을 극복하는 것이다.

3. 오픈소스 소프트웨어 기반 스몰베이직 코딩환경

이 장에서 본 연구에서 개발한 오픈소스 소프트웨어 기반 스몰베이직 코딩 환경 프로젝트, 마이스몰베이직(MySmallBasic)에 대하여 설명한다.

3.1 오픈소스 프로젝트 마이스몰베이직의 개요

오픈소스 프로젝트 마이스몰베이직은 자바로 작성한 스몰베이직 해석기, 표준 및 확장 라이브러리, 디버거를 포함한 편집 및 실행 환경으로 구성되어 있다. 그림 3은 마이스몰베이직을 활용하여 테트리스 프로그램을 실행하는 화면이다.

아래 깃허브(GitHub) 웹 사이트에 이 프로젝트의 소스코드, 문서, 스몰베이직 예제 프로그램들을 모두 내려받을 수 있다.

https://github.com/kwanghoon/MySmallBasic

마이스몰베이직 프로젝트는 총 2만여 라인의 자바 소스 프로그램으로, 스몰베이직 프로그램을 실행하는 해석기가 7,448라인, 스몰베이직 라이브러리가 13,445라인 규모에 해당한다.

이 오픈소스 프로젝트에 현재까지 15명의 개발자가 참여하여 소스 프로그램을 기여했다. 국내 2개 대학의 9명이 스몰베이직 해석기, 대부분의 표준 및 확장 라이브러리, 디버거를 포함한 GUI 코딩 환경을 개발하였고, 베트남 1개 대학의 6명이 확장 라이브러리를 개발하여 이 프로젝트에 기여하였다.

마이스몰베이직 프로젝트는 GPL(General Public License)를 채택하여 관련 소스 프로그램을 수정하였을 경우 공개하도록 의무화하였다. 그 이유는 마이스몰베이직 프로젝트를 누구나 다운받아 수정 가능하지만 기존의 스몰베이직 프로그램 동작과 다르게 변형된 프로젝트가 확산되는 것을 최대한 방지하기 위함이다. 2008년 처음 마이크로소프트 스몰베이직 코딩 환경이 공개된 이후로 상당히 큰 커뮤니티가 형성되었고 이 커뮤니티에서 상당한 양의 스몰베이직 프로그램들을 작성해왔다. 이러한 라이선스 정책을 통해 호환성을 유지하여 기존의 스몰베이직 코딩 커뮤니티가 활용할 수 있는 프로젝트로 진행하고자 한다.

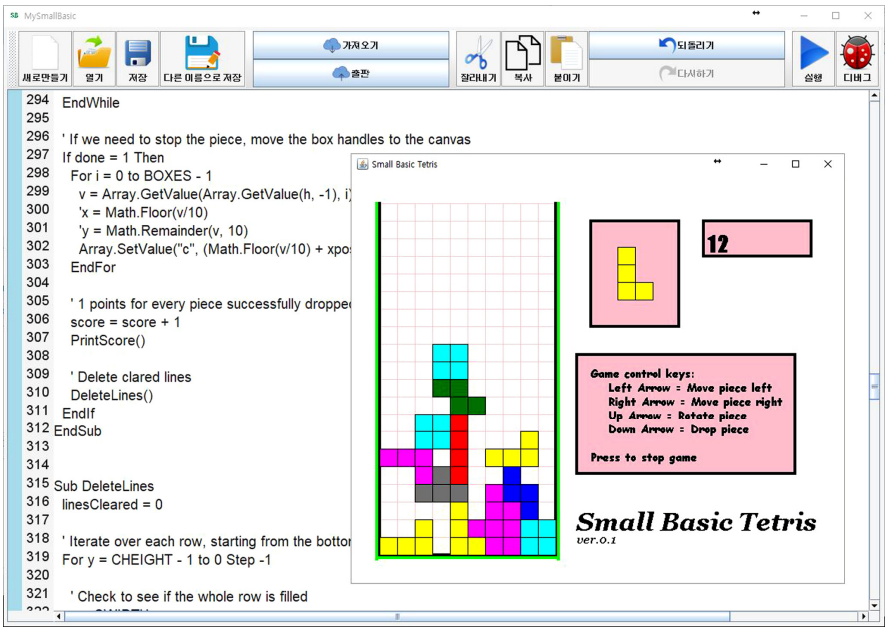

그림 3 마이스몰베이직 코딩 환경

Fig. 3 MySmallBasic-based Coding Environment

표 3 스몰베이직 벤치마크 프로그램
Table 3 Smallbasic Benchmark Programs

Name	Lines	Description
Bricks	243	Bricks game
CollisionPhysics	447	Physics simulation
Tetris	536	Tetris game
Sokoban	1165	Sokoban game

특히 스몰베이직 코딩 환경 간 호환성을 유지하기 위해 마이크로소프트에서 발행한 스몰베이직 프로그래밍 튜토리얼에 나열된 59개 예제 프로그램들을 활용하였다. 이 프로그램들은 최대 36라인, 전체 합계 470라인으로 비록 개별적으로 크기가 작지만 스몰베이직 프로그래밍 언어의 각 기능(If, For, While 등)을 검증하고 표준 라이브러리 함수들을 정확히 구현하였는지 확인하는데 충분하다. 이 예제 프로그램들을 마이스몰베이직에서 실행한 결과와 마이크로소프트 코딩 환경에서 실행한 결과가 모두 동일한 것을 확인하였다.

작은 튜토리얼 프로그램들과 함께 스몰베이직 커뮤니티에서 개발한 최대 천 라인 규모의 큰 예제 프로그램도 프로젝트에 포함시켜 테스트하였다. 표 3에 그 중 4가지를 사례를 보여준다.

이 논문의 첫 번째 저자는 소속 대학의 비전공 학생들을 대상으로 코딩 교육 <컴퓨터과학적사고> 과목의 실습에서 마이스몰베이직 프로젝트 기반 코딩 환경을 활용하였다. 논의 사항에서 이 경험에 대해 설명할 것이다.

코딩 교육을 받는 사용자 관점에서 마이크로소프트 스몰베이직과 마이스몰베이직의 주요 차이점은 다음과 같이 설명할 수 있다. 첫째, 마이스몰베이직은 자바로 구현하여 어느 운영체제에서나 실행할 수 있지만, 마이크로소프트는 닷넷 기반으로 윈도우에서만 실행할 수 있다. 둘째, 마이스몰베이직에서 제공하는 스몰베이직 디버거를 활용하여 프로그램 실행 과정을 자세히 살펴볼 수 있다. 셋째, 표준 스몰베이직 라이브러리 이외에도 햄스터 제어, SNS, 데이터베이스 등 확장 라이브러리를 활용하여 코딩 할 수 있다. 넷째, 마이크로소프트 스몰베이직에서 제공한 자동 완성 기능과 도움말 정보 덕분에 초보자가 많은 라이브러리 함수 이름과 쓰임새를 모두 기억할 필요가 없다. 다섯째, 마이크로소프트 스몰베이직 환경은 앱스토어와 연동하여 다른 사람이 작성한 프로그램을 쉽게 내려 받아 실행해볼 수 있다. 마지막 두 가지 사항은 마이스몰베이직을 개선할 사항이다.

3.2 스몰베이직 해석기

오픈소스 프로젝트 마이스몰베이직은 자바로 작성한 해석기를 통해 스몰베이직 프로그램을 실행한다. 스몰베이직 프로그래밍언어의 구문과 의미를 엄밀하게 설명하는 문서가 없기 때문에, 마이크로소프트 스몰베이직 코딩 환경에서 실행해보면서 그 구문과 의미를 역공학 분석하였다.

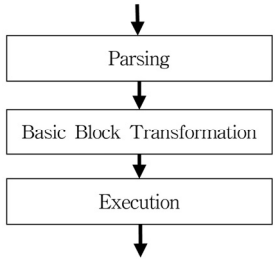

그림 4 해석기 실행 단계
Fig. 4 Execution Stages in Small Basic Interpreter

해석기는 그림 4와 같이 파싱 단계, 기본 블록으로 변환하는 단계, 실행하는 단계 순서로 실행된다.

3.2.1 파싱 단계

개발한 스몰베이직 파서의 토큰 분석(token/lexical analysis)은 40개의 토큰을 받아들이는 유한 오토마타(finite automata)를 설계하여 구현하였고, 구문 분석(syntax analysis)은 60개의 생산 규칙(production rules)로 구성된 LALR(1) 문법을 작성하여 적용하였다.

스몰베이직 파싱 단계에 필요한 토큰 분석과 구문 분석에 관한 명세는 프로젝트 웹 사이트에 공개하였다. 스몰베이직 파싱 단계에 필요한 이러한 명세 문서를 공개한 사례는 이전에 없었다.

스몰베이직 구문의 특성상 한 줄에 작성할 문장을 두 줄로 나누어 작성하면 구문 오류가 발생한다. 따라서 줄바꿈 문자를 단지 토큰을 구분하는 역할(delimiter)로만 사용하는 것이 아니라 구문 분석에 필요한 핵심 토큰으로 사용한다.

구현한 파서를 테스트하기 위해 3.1절에서 설명한 호환성 검증을 위한 59개 스몰베이직 프로그램과 4개 벤치마크 프로그램을 사용하였다.

3.2.2 기본 블록 변환 단계

기본 블록 변환 단계에서 스몰베이직 프로그램의 제어 흐름을 분석하여 기본 블록(basic block) 기반 제어 구조의 프로그램으로 변환하고 반복문 For와 While을 조건문 If, 라벨, 분기문 Goto로 대체한다. 예를 들어 그림 1의 예제 프로그램에 대해 이 변환을 거치면 그림 5의 프로그램이 된다.

각 기본 블록은 다음과 같은 공통 특성을 가지고 있다. 기본 블록은 항상 라벨로 시작한다. 그리고 기본 블록의 문장들 중 마지막에만 Goto문을 허용한다. 그림 5에서 기본 블록으로 변환된 프로그램 구조를 보면 $main과 $L0 라벨로 시작하는 두 블록들은 각각 마지막 문장이

```
$main:
    Goto $L0
$L0:
    i = 1
    Goto $L1
$L1:
  If i <= 20 then
    pic = Flickr.GetRandomPicture("Korea")
    GraphicsWindow.DrawResizedImage(pic,0,0,640,480)
    i = i + 1
    Goto $L1
  EndIf
```

그림 5 기본 블록 변환 단계 이후의 프로그램 구조
Fig. 5 The Program Structure After the Basic Block Transformation

Goto문이다. $L1의 블록에서 If 조건이 참인 경우 마지막으로 실행될 문장이 Goto문이고, If 조건이 거짓인 경우 If문이 마지막 문장이고 이후 프로그램 실행을 종료한다.

3.2.3 실행 단계

실행 단계에서는 시작 라벨로 약속한 $main의 블록에서 프로그램 실행을 시작해서 분기문으로 끝나는 블록들을 반복해서 실행한다. 마지막 문장이 더 이상 분기문이 아닌 블록을 실행하고 프로그램을 종료한다.

예를 들어, 그림 5의 예제에서 $main 라벨의 블록부터 실행하여 $L0와 $L1의 블록을 차례로 실행한다. 그런 다음 $L1의 블록 안에서 자신으로 분기하는 Goto문을 20회 반복 실행한 다음 If 조건이 거짓이 되면 분기할 블록이 더 이상 없기 때문에 프로그램을 종료한다.

이렇게 기본 블록들로 구조화된 프로그램을 실행하면 반복해서 분기하지만 다시 돌아오지 않는 패턴의 제어 흐름이 된다. 이 패턴의 제어 흐름을 구현하기 위해서 트램펄린(trampoline) 스타일[5]을 사용한다.

트램펄린 스타일은 아래의 드라이버 코드를 사용한다. 이 코드의 eval 함수는 변수 환경(env)과 실행할 블록을 인자로 받아 그 블록을 실행하고 다음 실행할 블록의 라벨을 반환한다. 이 라벨로 다음 실행할 블록을 찾아 다시 eval 함수로 반복 실행한다. 마치 트램펄린에서 점프하고 내려와서 다시 점프하는 과정을 반복하는 것과 유사하다.

```
currentLabel = "$main";
env = initialEnvironment();
while (currentLabel != null) {
  Block block = blockMap.get(currentLabel);
  label = eval(env, block);
}
```

블록 단위의 실행을 위해 라벨 이름을 기본 블록으로 매핑하는 자료 구조(blockMap)에 변환 단계를 거친 블록들을 가지고 있고, 스몰베이직 프로그램은 전역변수만을 허용하므로 변수 이름을 값으로 매핑하는 변수 환경(env)을 유지한다. 따라서 프로그램 시작 전에 변수 환경을 초기화하고 현재 선택된 (currentLabel이 가리키는) 블록을 실행하면서 이 변수 환경에 새로운 변수와 값의 쌍을 추가하거나 기존 변수의 값을 변경한다.

3.2.4 스몰베이직의 동적 타이핑

스몰베이직은 실행하면서 필요한 타입의 값으로 변환하여 사용하는 동적 타이핑을 이용한다. 스몰베이직 언어에서 문자열뿐만 아니라 정수와 실수를 포함한 숫자, 부울값, 심지어는 배열도 모두 문자열로 변환할 수 있고, 그렇게 표현된 문자열을 다시 필요한 타입의 값으로 변환할 수 있다.

이러한 동적 타이핑 특성을 명확히 이해하는 것은 스몰베이직 해석기를 구현할 때 매우 중요하지만 이에 대해 설명하는 문서가 없다. 이 논문에서 마이크로소프트 코딩 환경을 역공학 분석한 스몰베이직의 동적 타이핑을 표 4와 같이 정리한다.

각 타입의 값들을 어떻게 변환하는지 설명하기 전에 동적 타이핑이 적용되어 프로그램을 실행하는 예를 먼저 살펴보자.

- 식 "123" + 456은 문자열 "123"을 숫자 123으로 먼저 변환하고 숫자 456에 더하여 579 결과를 낸다.
- 식 "123" + "abc"는 "abc"를 숫자로 변환할 수 없기 때문에 "123"을 문자열로 간주하여 두 문자열을 붙여 "123abc"라는 결과를 낸다.
- 문장 If "123" Then stmt1 Else stmt2는 문자열 "123"은 부울값 "True"와 다르기 때문에 조건이 성립하지 않는다고 판단하여 stmt2를 실행한다.
- arr = "0=123;1=456;2=789"을 실행하여 변수 arr에 이 문자열을 넣으면 배열 원소 arr[0]은 문자열 "123"이 된다. 배열 arr의 원소 1과 2에도 유사하게 문자열 "456"과 "789"가 된다.

먼저 스몰베이직에서 숫자, 부울값, 문자열, 배열을 어떻게 작성하고, 각 타입의 값을 문자열로 변환하는 방법을 예를 들어 설명한다. 숫자는 123 또는 123.456로 작성하면 내부적으로 "123" 또는 "123.456"과 같이 숫자를 담고 있는 문자열로 변환한다. 부울값은 "True"와 "False"와 같이 대소문자 구분하지 않는 문자열로 작성한다.

일차원 배열은 키(첨자)와 값으로 표현한다. 앞에서 보인 예에서와 같이 첨자 0, 1, 2의 배열 원소가 123, 456, 789이면, 이 배열은 키와 값을 등호로 구분하고 각 배열 원소를 세미콜론으로 구분하는 형식의 문자열 "0=123;1=456;2=789"로 정의된다. 다차원 배열은 키(첨자)와 한 차원 낮은 배열의 (문자열로 변환된) 값을 매

표 4 스몰베이직 타입 변환 방법
Table 4 SmallBasic Type Conversion Rules

=>	number	string	boolean	array
number		(1)	(2)	(3)
string	(4)		(5)	(6)
boolean	(7)	(8)		(9)
array	(10)	(11)	(12)	

핑한 것으로 일반화하여 표현할 수 있다.

배열 첨자는 대소문자를 구분하지 않는 문자열로 정의한다. 따라서 숫자뿐만 아니라 문자열도 배열의 첨자로 사용할 수 있고, 심지어 문자열로 표현할 수 있는 배열도 첨자로 사용할 수 있다.

참고로 모든 변수와 배열의 원소는 빈 문자열 “”로 초기화되도록 정의되어 있다. 따라서 자바에서와 같이 널(Null) 값을 허용하지 않는다.

스몰베이직에서 타입 간 변환 방법을 설명하기 위해 프로그램으로 작성할 수 있는 4가지 타입(T)을 number, string, boolean, array(T)라 하자. 수직으로 배치된 타입을 가로로 배치된 타입으로 변환하는 방법을 설명한다.

(1) 123 또는 123.456을 “123” 또는 “123.456”으로 변환
(2) 모든 숫자를 거짓 “False”로 변환
(3) 모든 숫자를 원소가 없는 배열로 변환
(4) 숫자를 담고 있는 문자열 “123” 또는 “123.456”을 123 또는 123.456으로 변환. 그 외의 모든 문자열은 0으로 변환 (예: 빈 문자열 “”, “abc”)
(5) 대소문자를 구분하지 않고 문자열을 비교할 때 “True”와 동일하면 참, 아니면 거짓으로 변환
(6) 앞서 설명한 형식을 따르는 문자열을 키(첨자)와 값의 매핑인 배열로 변환. 이와 관련하여 아래에 제시할 보충 설명을 참고한다.
(7) 참과 거짓 모두 0으로 변환
(8) 참은 “True”와 거짓은 “False”
(9) 참과 거짓 모두 원소가 없는 배열로 변환
(10) 모든 배열은 0으로 변환
(11) 앞서 설명한 형식에 따라 배열을 문자열로 변환
(12) 모든 배열을 거짓으로 변환

앞에서 설명한 스몰베이직의 4가지 타입 간 변환 방법은 문맥에 독립적으로 정의한 것이다. 연산자에 따라 이 정의만으로 충분히 설명할 수 없는 타입 변환이 발생할 수 있다. 예를 들어, 1 + “True”에서 “True”를 숫자 타입으로 변환하면 1+0이 되고 문자열 타입으로 변환하면 “1True”가 된다. 마이크로소프트 스몰베이직은 이 경우 + 연산자를 숫자 더하기로 해석하기보다 문자열 붙이기로 해석한다. 마이스몰베이직도 이를 그대로 따르도록 해석기를 구현하였다. 만일 “True”를 숫자 0으로 변환하는 방법을 먼저 적용한다면 1 + 0이 되어 잘못된 결과를 낼 것이다. 이러한 차이가 발생할 가능성이 있기 때문에 동적 타이핑에 대한 정확한 정의가 필요하다.

반면에 1 - “True”의 경우 부울값 “True”를 숫자로 변환하기 때문에 앞에서 설명한 방법에 따라 1 - 0으로 변환한다. 마이너스 연산자뿐만 아니라 곱하기, 나누기 연산자도 동일한 변환 방식을 사용한다.

1 + arr과 같이 부울값이 아닌 배열의 경우도 비슷하다. + 연산자를 문자열 붙이기로 우선 해석하여 배열을 문자열로 변환하고 (그리고 숫자 1도 문자열로 바꾼 다음) 연산자를 적용한다. 마찬가지로 1 - arr의 경우 1 - 0으로 바꾸어 진행한다.

크기를 비교하는 연산자(<, >, <=, >=)는 숫자에 대해서만 정의되어 있다. 따라서 “abc” < “abcdef”를 사전식 순서로 생각하면 참이 되지만 스몰베이직에서 양쪽 문자열을 숫자로 바꾸면 모두 0이 되므로 0 < 0은 거짓이다.

같다를 비교하는 연산자(=)는 문자열로 변환한 다음 이 문자열이 같으면 참이라고 판단하도록 정의되어 있다.

마지막으로 배열을 문자열로 변환하는 과정에서 아주 미묘한 상황이 발생할 수 있음을 설명한다. 앞서 사용한 배열을 표현한 올바른 문자열 예는 “0=123;1=456;2=789”이다. 만일 어떤 이유에서 세 번째 등호가 빠진 “0=123;1=456;2789” 문자열을 배열 변수 arr에 대입하면 그 결과는 무엇일까? 형식 오류가 발생했기 때문에 아무 원소도 없는 빈 배열이 될 수도 있고, 또는 형식이 올바른 첨자 0과 1의 원소만 포함하고 그 이후 “2789”를 무시한 원소가 2개 있는 배열이 될 수도 있다. 마이크로소프트 스몰베이직에서 후자의 방법으로 정의하고 있음을 확인하였다. 이렇게 사소하지만 미묘한 타입 변환의 차이가 프로그램 실행 결과에 영향을 준다. 따라서 명확한 정의가 있어야 서로 다른 스몰베이직 코딩 환경 간 실행 결과가 동일하도록 보장할 수 있다.

3.3 스몰베이직 해석기 확장

앞 절에서 기존의 스몰베이직 코딩 환경과 호환되는 스몰해석기를 만들기 위해 필요한 주제들을 논의하였다. 이 절에서 이 호환성을 최대한 유지하면서 언어, 라이브러리, 코딩 환경을 확장 개발한 내용을 설명한다.

3.3.1 확장 라이브러리

마이스몰베이직의 오픈소스 프로젝트에 외부 개발자가 참여하여 쉽게 기여할 수 있는지 확인할 목적으로 확장 라이브러리를 시험 개발하였다. 커뮤니티에서 다양한 확장 라이브러리를 개발하면 학생들의 흥미를 높이는 코딩 교육을 기획하고 진행하는데 도움이 될 것이다. 기존 스몰베이직 표준 라이브러리에 10가지 새로운 라이브러리를 추가로 확장 개발하여 목표하는 바를 검증하였다.

외부 개발자가 이 프로젝트에 참여하여 새로운 라이브러리를 개발하고 추가하더라도 이미 개발된 스몰베이직 해석기와 코딩 환경을 변경하지 않도록 독립적인 라이브러리 구조를 설계할 필요가 있다.

기본적인 구조는 다음과 같이 요약할 수 있다. 각 사항에 대해 이후 자세히 설명한다.

- 스몰베이직의 각 라이브러리는 자바 클래스이다.
- 라이브러리 함수는 이 클래스의 멤버함수로, 라이브러리 변수는 이 클래스의 멤버변수이다.
- 스몰베이직의 (라이브러리에서 다루는) 숫자, 문자열, 부울, 배열의 값을 자바 클래스 Value를 기반 클래스로 하는 NumberV, StrV, BooleanV, ArrayV의 객체로 표현한다.

기본적으로 새로운 라이브러리는 동일한 이름의 자바 클래스로 작성하고 라이브러리 클래스들을 모아놓은 자바 패키지에 포함시킨다. 스몰베이직 프로그램에서 이 라이브러리의 함수와 변수를 이 클래스의 정적 메소드와 정적 멤버 변수로 구현하다. 이러한 방식으로 라이브러리를 위한 자바 클래스를 만들면 자바 리플렉션(Reflection) 방법으로 라이브러리 함수와 변수를 접근하도록 구현할 수 있다. 예를 들어, 스몰베이직 해석기에서 그림 1의 GraphicsWindow.DrawResizedImage 함수 호출을 구현하는 방법은 문자열 “GraphicsWindow”와 “DrawResizedImage”를 자바 리플렉션 방법으로 해당 클래스와 메소드를 각각 찾아 호출한다. Graphics Window.Width와 같은 라이브러리 변수를 읽고 쓸 때도 마찬가지로 라이브러리 이름의 문자열과 변수 이름의 문자열을 자바 리플렉션 방법을 통해 실제 라이브러리 변수를 참조해 읽기와 쓰기를 진행한다.

라이브러리 변수를 다룰 때 라이브러리 함수를 호출하는 것과 다른 점이 있다. 스몰베이직 프로그램에서 GraphicsWindow.Width에 폭을 지정하면 윈도우 크기가 바로 변경된다. 이 동작을 지원하기 위해서 라이브러리에 반드시 notifyFieldAssign 함수를 포함시켜 프로그램 해석기에서 이 라이브러리의 변수를 갱신한 다음 이 함수를 호출하여 갱신한 변수 이름을 알려주도록 설계하였다. 이 함수가 호출되면 인자로 받은 갱신된 변수 이름을 확인하고 윈도우 크기를 변경하는 액션을 취할 수 있다. 라이브러리 변수를 읽기 전에는 notifyFieldRead 함수를 두어 프로그램 해석기에서 읽을 변수 이름을 전달하면 라이브러리에서 이 변수의 값을 미리 설정해둔다. 예를 들어, 스몰베이직 프로그램에서 날짜를 확인할 때 해석기에서 Clock.Date 변수를 읽기 전에 Clock 라이브러리의 notifyFieldRead를 호출하여 현재 날짜로 설정해둔 다음 이 변수를 읽는다.

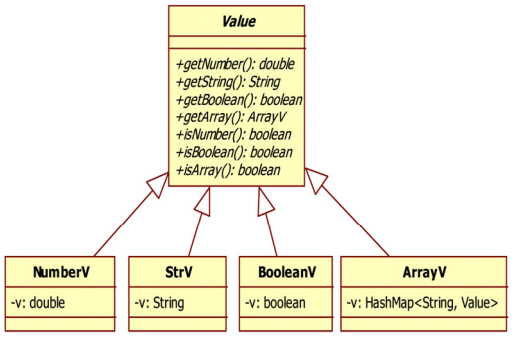

그림 6 스몰베이직 값을 표현한 자바 클래스
Fig. 6 Java Classes for SmallBasic Values

라이브러리를 사용할 때마다 자바 리플렉션을 사용하는 것이 다소 비효율적일 수 있으나 코딩 교육의 목표로 작성한 프로그램의 경우 학생들에게 전혀 방해 요소가 되지 않는다. 마이스몰베이직에서 실행한 프로그램이 느려 실습 진행이 어려운 적은 없었다. 향후 연구로 해석기 구조를 컴파일러 구조로 발전시켜 스몰베이직 프로그램으로부터 자바 프로그램을 생성하면 자바 리플렉션을 사용하지 않고 라이브러리를 사용할 수 있을 것이다.

스몰베이직 라이브러리 함수와 변수를 자바로 구현할 때 값(value)을 자바로 표현하는 방법은 마이스몰베이직 해석기의 구현이 변경되더라도 동일하게 유지되어야 한다. 이 목적을 위해 그림 6과 같은 계층의 자바 클래스들을 정의하였다.

마이스몰베이직에서 다루는 값의 숫자, 문자열, 부울값, 배열 타입을 Value 클래스와 이를 상속받아 정의한 4개의 클래스들로 표현한다. Value 클래스는 3.2.4절에서 설명한 (문맥에 독립적인) 동적 타이핑을 지원하는 메소드들을 가지고 있다. 예를 들어 숫자를 표현하는 객체가 주어져 있고 숫자를 문자열로 변환하려면 (동적 타입 변환 (1)의 경우) 이 객체의 getSring() 메소드를 호출한다.

앞에서 설명한 방식의 라이브러리 구조에 맞추어 작성한 확장 라이브러리 사례는 표 5와 같다. Assert, List, Tree, Graph 라이브러리는 처음부터 새로 개발하였고, Chart, Database, Facebook, Hamster, Video, Weka 라이브러리는 관련 오픈소스 소프트웨어를 활용하여 개발하였다.

예를 들어, 햄스터 로봇 제어 라이브러리인 Hamster는 코딩 교육에 피지컬 컴퓨팅[6,7] 개념을 도입하여 흥미를 높이기 위해서 추가되었다. 햄스터 로봇의 모터를 스몰베이직 프로그램을 통해 제어하고 부착된 센서 정보를 받아 상황에 따라 상호작용한다. 햄스터 라이브러리는 모터 속도를 제어하는 Wheel 함수와 방향을 전환하는 Rotate함수, 장애물을 감지하는 IsObstacleInDistance 함수를 포함하여

표 5 확장 라이브러리
Table 5 Extended Libraries

Library	Description
Assert	Assertion functions
Chart	Chart drawing functions
Database	SQLite3-based Database functions
Facebook	RestFB-based Facebook functions
Graph	Graph functions
Hamster	Hamster Robot control functions
List	List functions
Tree	Tree functions
Video	Video player functions
Weka	Weka-based learning functions

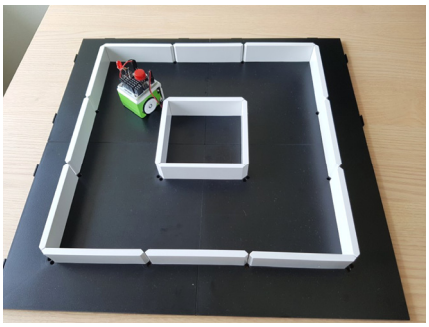

그림 7 스몰베이직 프로그램으로 제어하는 햄스터 로봇
Fig. 7 Hamster Robot Controlled By SmallBasic Program

71개의 함수로 구성되어 있다. 이 라이브러리에 대한 설명과 구현 방법에 대한 자세한 내용은 [8]을 참고한다.

그림 7은 확장 라이브러리를 사용한 스몰베이직 프로그램으로 햄스터 로봇을 제어하여 장애물을 만나면 오른쪽으로 회전하는 데모를 보여준다. 이 프로그램을 사용하여 햄스터로 하여금 미로를 탈출하도록 하는 초보자의 흥미를 끄는 코딩 예제를 그림 8과 같이 만들었다.

이 외에도 데이터마이닝 라이브러리 Weka 오픈소스 소프트웨어를 기반으로 학습 함수를 제공하는 확장 라이브러리를 만들어 스몰베이직으로 인공지능 틱택토 프로그램을 작성하였다[9]. 그리고 Database와 Facebook 라이브러리는 베트남 HUST 대학 2팀(총 6명)이 GitHub 프로젝트 사이트에 제공하는 라이브러리 구조 문서를 보고 개발하여 기여한 확장 라이브러리이다.

이와 같이 확장 라이브러리들을 개발한 경험을 바탕으로 새로운 라이브러리를 개발하고 추가하더라도 이미 개발된 스몰베이직 해석기와 코딩 환경에 영향을 주지 않는 독립적인 라이브러리 구조를 갖추었음을 확인하였다. 이러한 구조를 활용하면 누구나 마이스몰베이직 오픈소스 소프트웨어 프로젝트에 새로운 라이브러리를 쉽게 기여할 수 있을 것이다.

```
While “True”
 Hamster.Wheel(40)
 If Hamster.IsObstacleInDistance(35) = “True” Then
  Hamster.Rotate(30)
  While “True”
   If Hamster.IsObstacleInDistance(35)=“False” Then
    Goto ESCAPE
   EndIf
  EndWhile
 EndIf
ESCAPE:
EndWhile
```

그림 8 스몰베이직 프로그램 예제: 햄스터 로봇 제어
Fig. 8 A Small Basic Program for Hamster Robot Control

3.3.2 코딩 환경 확장: 소스 프로그램 디버거

마이크로소프트 스몰베이직 코딩 환경에서 디버깅 기능을 제공하지 않아 초보자가 스몰베이직 프로그램의 제어 흐름이나 특정 시점의 변수 값을 확인하는데 어려움이 있었다. 디버거를 통해 스몰베이직 프로그램을 더 잘 이해할 수 있도록 환경을 제공하기 위해 마이스몰베이직 프로젝트에서 소스프로그램 수준의 디버거를 개발하였다. 코딩 환경에서 소스 프로그램의 원하는 위치에 정지점을 설정하고 디버깅 모드로 실행하는 방법을 제공한다. 한 줄씩 실행하거나 정지점까지 실행하는 방법을 제공하고, 실행하면서 변경되는 변수 값을 코딩 환경에서 직접 확인할 수 있는 기능을 제공한다. 자세한 내용은 [10]을 참고한다.

오픈소스 프로젝트로 개발하였기 때문에 디버거 개발과 같은 코딩 환경의 확장이 용이하였다. 왜냐하면 디버거 개발을 위해 마이스몰베이직 프로젝트의 해석기 내부 구조를 반드시 알아야 하기 때문이다. 특히 구문 분석(Parsing)에서 얻은 추상구문트리(abstract syntax tree와 변수 환경에 관한 정보는 디버거를 구현할 때 반드시 필요하다.

3.3.3 스몰베이직 언어의 확장: Assert 문

오픈소스 소프트웨어 프로젝트의 속성상 다수의 개발자가 코드를 기여할 때마다 CI 기반으로 빌드하고 테스트할 필요가 있다. 마이스몰베이직의 코딩 환경은 스몰베이직 프로그램들을 활용하여 회귀테스트를 한다. 새로운 코드가 프로젝트에 추가될 때마다 이 과정을 자동화하기 위해 프로그램에 assert 문을 작성하는 방법을 구현하였다.

```
a = File.WriteContents("tmp.txt", "abcd")
b = File.ReadContents("tmp.txt")
```

```
c = File.WriteContents("tmp.txt", "\r\nabcd")
d = File.ReadContents("tmp.txt")

'@assert a = "SUCCESS"
'@assert b = "abcd"
'@assert c = "SUCCESS"
'@assert d = "\r\nabcd"
```

스몰베이직에서 작은따옴표로 시작해서 줄 바꿈 문자까지 주석에 해당한다. 테스트 담당자가 주석에 확장된 키워드 @assert와 조건식을 작성하면 마이스몰베이직 해석기가 이 주석이 있는 줄을 자동으로 Assert 라이브러리의 함수를 호출하는 "assert(조건식)"으로 바꾼다. 이 줄이 실행될 때 이 조건식을 계산하여 참이 되면 다음 줄로 이동하고 거짓이면 그 위치에서 에러 메시지를 출력하고 실행을 종료한다. 위 프로그램에서 파일을 읽고 쓰기를 두 차례씩 반복하면 변수 a, b, c, d에 지정한 문자열이 저장되어 있어야 한다. 이를 assert문으로 작성하였다.

주석으로 assert문을 작성하기 때문에 기존 마이크로소프트 코딩 환경에서 assert문을 포함한 프로그램을 수정하지 않고도 그대로 실행할 수 있다.

4. 논의 사항

4.1 코딩 교육 활용 사례

이 절에서 마이스몰베이직을 컴퓨터과학적사고 과목에서 실습 용도로 활용하였을 때 얻은 경험을 설명한다. 이 프로젝트는 아직 초기 단계이므로 코딩 교육의 효과를 정량적으로 판단하기에 부족하다. 따라서 이 절에서는 활용 경험을 정성적으로 서술한다.

컴퓨터과학적사고 과목의 주요 주제는 컴퓨터과학적사고, 정보표현, 논리, 기초 프로그래밍, 기초 자료구조 및 알고리즘이다. 표 6에 이 과목에서 한 학기 동안 다룬 5가지 주제를 요약하였다.

표 6에서 나열한 주제의 강의를 진행하면서 전반부는 주로 종이와 연필을 활용하여 실습을 진행하였고 후반부에 특히 4번째 주제 이후부터 본격적으로 스몰베이직을 활용한 코딩 실습을 진행하였다.

각 주제에 대해 아래 표 7의 스몰베이직 프로그램을 학생들로 하여금 마이스몰베이직을 사용하여 전체를 작성하거나 또는 일부를 채워 완성하는 방식으로 실습을 진행하였다.

마이스몰베이직을 컴퓨터과학적사고 강의 및 실습에 적용한 결과 다음과 같은 흥미로운 사항을 발견할 수 있었다. 첫째, 스몰베이직이 배우기 쉬운 프로그래밍언어라는 장점을 다시 한 번 확인할 수 있었다. 강의 시간에 스몰베이직을 직접 가르치는 것을 지양하였고 튜토리얼 문서를 제공하고 스스로 공부하도록 하였다. 다만 강의 시간에 스스로 공부하였는지 확인하는 퀴즈를 진행하였다.

표 6 컴퓨터과학적사고 과목의 주요 주제
Table 6 The Five Main Topics in the Computational Thinking Lecture

	Topics
1	Computational Thinking - What is computational thinking - Understanding effective procedures
2	Information Representation - Bits - Number, String
3	Logic - Logic application in daily lifes - Symbolic logic and inference rules
4	Computer Programming - Elements of programming languages - How to design effective procedures
5	Information Organization - List, Tree, Graph - Algorithms

- For나 While과 같은 반복문을 작성하는 것은 어려워하지만 If와 Goto문에 매우 쉽게 적응하는 것을 알 수 있었다. 정식 프로그래밍 과목이라면 전자의 반복문을 사용하는 것이 바람직하지만 그렇지 않다면 If와 Goto만을 사용하여 제어 흐름을 구성하는 것도 가능한 방법이라 판단된다.
- 전역변수만을 사용하되 서브루틴의 인자를 두지 못하도록 설계한 스몰베이직의 특징 때문에 학생들은 서브루틴을 자동으로 돌아오는 분기문 Goto로 매우 쉽게 이해할 수 있었다. 다른 프로그래밍언어의 경우 인자 전달에 대한 설명을 했어야 하겠지만 마치 클래스와 객체 개념과 같이 초기 코딩 교육에는 필요한 요소는 아니다.

둘째, 학생들의 흥미를 끌기 위해 스몰베이직 표준 라이브러리로 충분하였다. 실습에서 주로 그래픽스 라이브러리를 많이 사용하였고 그 다음으로 텍스트 라이브러리를 사용하였다. 리스트, 트리, 그래프에 대한 확장 라이브러리를 새로 만들었지만 배열로 각 자료 구조를 표현하는 방법을 학생들이 직관적으로 잘 이해하였기 때문에 별도의 확장 라이브러리들을 활용하지 않았다.

셋째, 최소의 특징만을 고려한 스몰베이직의 경우조차도 프로그래밍 요소(If, For 등)와 라이브러리 함수와 변수가 너무나 다양하기 때문에 초보자가 어느 것을 사용해서 프로그램을 작성해야 하는지 막막하게 생각하는 경우가 빈번하게 발생한다. 이 문제에 대한 대응 방안이 필요하다. 학생들이 스몰베이직 프로그램을 작성할 때

표 7 컴퓨터과학적사고 과목에서 설계한 스몰베이직 프로그램 예제

Table 7 A Design of Smallbasic Programs for the Computational Thinking Lecture

	SmallBasic Programs	Related Topics
1	Newtwon Square Root Calculation	Computational Thinking
2	Rock-Scissors-Paper Game	Information Representation
3	Course Registration Data Analysis	Basic Programming
4	Tic-tac-toe	How to design an effective procedure
5	Rush Hour Game	Information Representation
6	Josephus Problem	List
7	Word Auto Completion	Tree
8	Uncovering a Hidden Map	Graph
9	Finding Subway Directions	Dijkstra's Shortest Path Algorithm

선택할 수 있는 범위를 한정지어 주는 방법이 있다면 더 쉽게 작성할 수 있을 것이다. 예를 들어 지하철 길찾기 프로그램의 경우 최단거리 찾는 알고리즘의 핵심만 구현하면 나머지 그래픽을 다루는 요소는 미리 제공하는 방식을 사용하였다.

넷째, 마이스몰베이직 프로젝트는 초기 개발 단계이므로 구문 분석 단계나 실행 단계에서 오류가 발생하는 경우 코딩 환경의 사용자에게 보다 쉽게 이 오류를 해결할 수 있도록 자세히 설명하는 방법을 아직 제공하지 못하고 있다. 초보자가 사용하는 코딩 환경을 목표로 하는 만큼 오류 처리에 대한 많은 고려가 필요하다.

5. 결론 및 향후 연구

이 논문의 연구로 개발한 오픈소스 소프트웨어 프로젝트 마이스몰베이직을 소개하였다. 마이스몰베이직은 마이크로소프트 코딩 환경에서 개발한 스몰베이직 프로그램을 그대로 실행할 수 있도록 설계한 교육용 코딩 환경이다. 자바 기반으로 개발하였기 때문에 윈도우, 리눅스, 맥 운영체제에 모두 실행가능하다. 새로운 10가지 라이브러리를 개발하고, 새로운 소스 수준의 디버거를 개발해 봄으로써 커뮤니티에 기여할 수 있는 인프라를 갖추었음을 확인하였다. 아울러 스몰베이직 프로그래밍 언어의 파서와 동적 타이핑 의미에 대한 명세를 처음으로 문서화하였다.

향후 연구로, 4장에서 논의한 바와 같이 효과적인 코딩 교육을 위한 환경이 되기 위해서 사용자와 상호작용하는 더욱 다양한 기능이 필요하다. 마치 스크래치에서 사용가능한 블록들이 한정되어 있듯이 작성할 수 있는 범위를 한정지어 제시하는 코딩 환경이 사용자에게 도움이 될 것이다. 또한 구문 오류나 논리 오류가 발생한 경우 스스로 그 이유를 파악하고 대응할 수 있는 환경을 제공해야 한다. 이러한 다양한 아이디어를 적용할 수 있는 코딩 교육 플랫폼으로써 오픈소스 소프트웨어 프로젝트 마이스몰베이직을 활용할 계획이다.

두 번째, 마이스몰베이직을 실제 코딩 교육에 적용한 사례를 정량적으로 측정하여 코딩 교육 효과를 평가하는 연구가 필요하다. 다른 교육용 프로그래밍 언어 및 환경을 사용했을 때 교육효과와 비교하는 연구도 필요하다.

References

[1] Microsoft Small Basic [Internet], http://smallbasic.com.

[2] P. Conrod, L. Tylee, The Developer's Reference Guide To Microsoft Small Basic, Kidware Software LLC, 2010.

[3] M. Hall, E. Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten, *The WEKA Data Mining Software: An Update, SIGKDD explorations*, Vol. 11, No. 1, 2009.

[4] Open Source Small Basic [Internet], https://github.com/kwanghoon/MySmallBasic.

[5] G., Steven E. and F., Daniel P. and Wand, Mitchell, Trampolined Style, *Proc. of the Fourth ACM SIGPLAN International Conference on Functional Programming (ICFP 1999)*, pp. 18-27, Paris, France, 1999.

[6] D. S. Seo, "Concept and Technical Basics of Physical Computing," *Korea Society of Design Science 2006 Fall Conference Proceeding*, pp. 270-271, 2006.

[7] J. Park, A Study on Application of STEAM education with Robot in Elementary School, *Journal of the Korea Society of Computer and Information*, Vol. 17, No. 4, pp. 19-22, Apr. 2012.

[8] S. Y. Park, M. Y. Jo, K. H. Choi, "A Study on the Design and Implementation of SmallBasic Library for Educational Robot Programming," *Korea Information Processing Society Spring Conference*, Vol 24, No. 1, pp. 399-402, Jeju National University, Apr. 2017.

[9] J. Y. Kim, S. W. Jeong, S. M. Jo, K. H. Choi, "A Study on the Design and Implementation of SmallBasic Library for Educational AI Programming," *Korea Information Processing Society Spring Conference*, Vol. 24, No. 1, pp. 694-696, Jeju National University, Apr. 2017.

[10] M. Y. Jo, K. H. Choi, "A Study on Design and Implementation of Debugger on SmallBasic Programs," *KIISE Korea Software Congression 2017*, pp. 2195-2197, Busan Bexco, December 20-22, 2017.

최 광 훈
정보과학회논문지
제 24 권 제 2 호 참조

김 가 영
2018년 전남대학교 전자컴퓨터공학부 소프트웨어공학전공(공학사). 2018년~현재 전남대학교 전자컴퓨터공학과 석사 과정. 관심분야는 프로그래밍언어, 컴파일러, 소프트웨어공학

창 병 모
1988년 서울대학교 컴퓨터공학(공학사) 1990년 KAIST 전산학과(공학석사). 1994년 KAIST 전산학과(공학박사). 1995년~현재 숙명여자대학교 소프트웨어학부 컴퓨터과학전공 교수. 관심분야는 프로그래밍언어, 프로그램 분석 및 검증, 소프트웨어 보안