스몰베이직 언어의 동적 타이핑 구조 분석 및 표준 라이브러리 개발에 관한 연구*

김가영 최광훈
전남대학교 전자컴퓨터공학부
kirayu15@gmail.com kwanghoon.choi@jnu.ac.kr

A Study on Dynamic Typing and Development of Standard Library in Small Basic Programming Language

Gayoung Kim Kwanghoon Choi
Department of Electrocis and Computer Engineering, Chonnam National University, Gwangju

요 약

본 연구는 다양한 운영체제와 플랫폼에서 스몰베이직 프로그래밍을 할 수 있는 환경을 제공하고, 라이브러리를 사용하고 확장하는 것을 목표로 진행 중인 연구 내용을 요약한다. 마이크로소프트 스몰베이직은 처음 컴퓨터 프로그래밍을 배우는 사람에게 쉽게 다가갈 수 있는 프로그래밍 언어이다. 단 14개의 키워드로 구성되어 있어 매우 간단하고 배우기 쉽다. 하지만 프로그래밍 환경을 윈도우즈 운영체제에서만 사용해야하는 단점이 있다. 소스 코드가 공개되어 있지 않아 새로운 요구사항을 맞추기도 어렵다. 이러한 문제점을 해결한 다양한 운영체제에서 사용이 가능하며 라이브러리 확장이 가능한 스몰베이직 환경을 설계하고 구현하고자 한다. 또한 마이크로소프트 스몰베이직의 동적 타이핑 구조에 대한 분석을 통해 명확한 규칙을 설명하고자 한다. 다양한 운영체제에서 스몰베이직 프로그램을 작성하고 PC 뿐만 아니라 안드로이드 기반 스마트폰에서 실행할 수 있다.

1. 서 론

모바일 시대를 넘어서 다가오는 사물인터넷 시대에는 모든 사물에 컴퓨터 기능이 탑재되고 이를 운용하기 위한 소프트웨어가 필요하다. 그만큼 사회 전반에 소프트웨어의 역할이 커지게 되어 소프트웨어에 대한 이해가 매우 중요하다. 대학에서 컴퓨터 이외의 전공자에게도 코딩 교육을 시키고 있고, 초중고교에서 일정시간의 코딩 교육이 의무화되고 있는 것이 세계적인 추세이다.

마이크로소프트의 비제에 라지(Vijaye Raji)가 개발한 스몰베이직(Small Basic)은 처음 코딩을 접하는 사람을 위한 윈도우 기반 프로그래밍 언어 및 코딩 환경을 제공한다. 코딩을 전혀 배우지 않은 사람이 처음 배우기에 적합하도록 이해하기 쉬운 최소한의 언어 요소만 도입하여 만들었다. 코딩 환경이 매우 단순하여 적응하기 쉽고, 라이브러리를 활용하여 텍스트 및 그래픽스 프로그램을 쉽게 작성할 수 있다. 또한 소스 프로그램을 공유할 수 있는 일종의 앱스토어를 제공한다.

스크래치나 앱인벤터와 같은 교육용 코딩 환경을 통해서 코딩의 기초 개념을 배울 수 있지만 이러한 환경에서는 코드가 없는(그림으고 작성된) 프로그램을 통해서 코딩을 배운다. 스몰베이직은 이러한 점에서 다르게 접근한다. 파이썬이나 C/C++/Java는 초보자가 처음 코딩 개념을 배우기에 너무 많은 특징을 보유하고 있어 장벽이 될 수 있다. 이러한 점에서 스몰베이직은 처음 코딩을 배우기에 적합한 프로그래밍 언어이다.

그럼에도 불구하고, 스몰베이직은 마이크로소프트의 닷넷 프레임워크에 종속되어 리눅스, 맥 운영체제, 웹, 안드로이드 스마트폰에서 실행하기 어렵고, 스몰베이직의 확장 역시 닷넷프레임워크와 연관된 도구를 통해 개발해야 한다.

본 연구가 기여하는 바는, 첫째 스몰베이직을 기존의 윈도우 운영체제뿐만 아니라 리눅스와 맥에서도 활용할 수 있는 코딩환경과 표준 라이브러리를 오픈소스 소프트웨어로 개발한 것이다. 둘째 마이크로소프트 스몰베이직에서 명확히 문서화 되지 않은 프로그램의 동적 타이핑 규칙을 분석하여 그 구조를 제시한다.

2. 관련 연구

스몰베이직을 이용해 플리커 웹사이트에서 지정한 주제의 사진들을 찾아 컴퓨터 바탕 화면에 슬라이드 쇼하는 프로그램[2]을 [그림 1]과 같이 간단하게 작성할 수 있다.

* 이 논문은 2017년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2017R1A2B4005138).

```
For I=1 To 20
 pic = Flickr.GetRandomPicture("Korea")
 Desktop.SetWallPaper(pic)
EndFor
```

(그림 1) A Small Basic Program for Wall Paper Slide Show

스몰베이직의 특징을 다음과 같이 요약할 수 있다.

- 스몰베이직의 키워드는 전체 14개, IF, Then, Else, EndIf, While, EndWhile, ElseIf, For, To, Step, EndFor, Goto, Sub, EndSub이다.
- 특별한 선언 없이 변수를 사용하고, 스코프(scope) 개념이 없이 모든 변수를 전역 변수로 사용한다.
- 타입 개념 역시 없다. 문자열 상수와 숫자 상수를 프로그램에서 사용할 수 있으나 내부적으로 모두 문자열로 관리하고 연산을 수행할 때 적절히 해석한다. 배열을 지원한다.
- 직관적으로 이해하기 쉬운 Goto문과 서브루틴 호출을 통한 제어 흐름 방식을 지원한다. IF, For, While을 지원한다.
- GraphicsWindow나 TextWindow과 같이 간단한 형태의 객체 개념을 통해 라이브러리 함수와 설정 변수를 제공한다.

스몰베이직과 코딩 교육에 사용되는 다른 언어나 환경을 다음 표와 같이 간략하게 비교할 수 있다.

<표 1> The Comparison for Coding Education Language

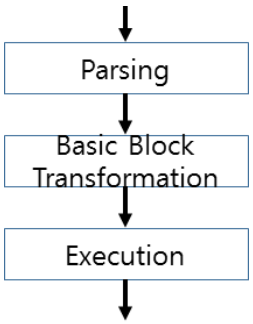
	Simplicity	Accessibility	Library
Scratch AppInventor	Simple (Codeless)	Web/PC	Limited
Small Basic	Simple	Windows	Limited
Python C/C++/Java	Complex	OS-neutral	Rich

배우기 쉬운 장점이 있음에도 불구하고 기존 마이크로소프트 스몰베이직 환경은 윈도우 운영체제에서만 사용이 가능한 문제가 있었다. 또한 스몰베이직 언어의 의미(Semantics)를 명확히 정의한 문서가 없어 스몰베이직 환경을 개발할 때 동적 타입 규칙을 파악하기 어렵다는 단점이 있다.

이러한 문제점을 해결하고자 이 연구에서 스몰베이직 프로그래밍 환경과 표준 라이브러리를 Java 기반으로 개발하여 윈도우 운영체제의 의존성을 없애고, 스몰베이직 언어를 정확히 구현하기 위한 동적 타이핑 규칙을 분석하여 타입에 대한 명확한 규칙을 제시하고자 한다.

3. 스몰베이직 프로그램 해석기

2절에서 설명한 동기로 Java 기반 스몰베이직 언어의 환경을 다음 설명과 같이 개발하였다. 스몰베이직 환경은 크게 해석기 부분과 표준 라이브러리 부분으로 구성할 수 있다. 첫째, 해석기 부분은 [그림 2]와 같이 파싱 단계, 기본 블록으로 변환하는 단계, 실행하는 단계로 나누어 구현하였다.



(그림 2) The Internal Stages of the Small Basic Interpreter

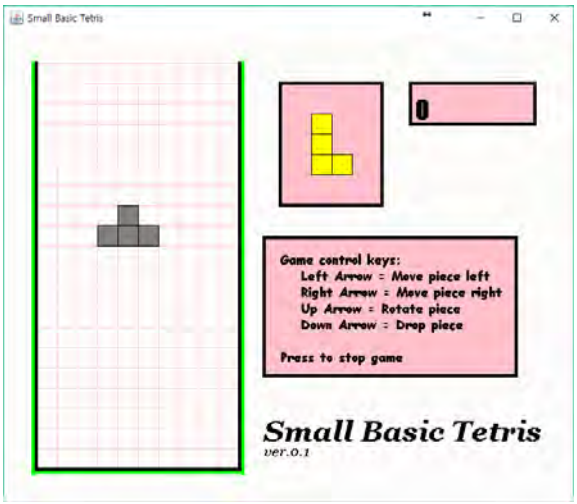
둘째 스몰베이직 표준 라이브러리를 Java 기반으로 모두 구현하였다. 스몰베이직은 20개의 표준 라이브러리를 가지고 있으며, 콘솔창에서의 입출력, 그래픽 창에 도형 그리기 등을 라이브러리를 통해 쉽게 이용할 수 있다.

<표 2 > Standard Library of Small Basic

Library	Description
Array	Array functions
Clock	System Clock functions
Controls	Button, TextField, Multi Line TextField
Desktop	Desktop WallPaper Setting
Dictionary	Onlinie Dictionary Service
File	File and Directory management
GraphicsWindow	variable Graphics functions
ImageList	Image management
Math	Math method
Mouse	Mouse Cursor and Button Properties
Network	File download
Program	Program execute Controls
Shapes	Graphics figure
Sound	Sound play, stop, pause function
Stack	Stack functions
TextWindow	Text Input/Output in Console
Text	Text Calculation
Timer	Timer functions
Turtle	Turtle Graphics

현재 MySmallBasic 해석기는 기존 마이크로소프트 스몰베이직에서 실행되던 500라인 가량의 테트리스와 물리 시

뮬레이션 프로그램을 동일하게 동작시킬 수 있도록 구현했다.



(그림 3) Tetris Program Running on MySmallBasic

마이크로소프트에서 제공하는 스몰베이직 튜토리얼 문서에서 발췌한 26가지 프로그램을 개발한 MySmallBasic 환경에서 동작하게 만들었다. 현재 이 환경에서 240~500라인 규모의 실제적인 스몰베이직 프로그램 예제인 테트리스, 물리시뮬레이션, 벽돌깨기 게임이 잘 동작함을 확인했다.

<표 3> Sample Program List in Small Basic

HelloWorld	FontYellowColor	Variables	If
Goto	For	ForStep	While
GraphicsWindow	GraphicsWindowConfig	DrawLine	LineColor
LineThickness	Rectangle	Ellipse	Circle
Random	Fractal	Subroutine	Array
ArrayIndex	MultiDimArray	Event	Events
Flickr			

현재 1200라인 규모의 Sokoban 프로그램 예제를 동작할 수 있도록 개발 중이다. 이 프로그램은 애니메이션 효과, 웹에서 이미지·소리 등의 리소스 파일을 다운받는 기능, 사운드 기능 등이 복합된 매우 복잡한 프로그램이다.

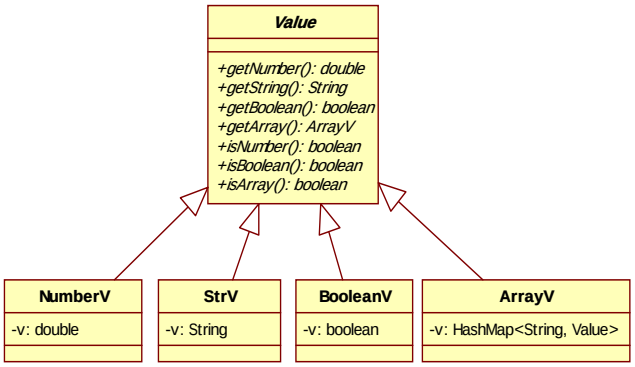
4. 논의 사항

스몰베이직 개발환경을 구현할 때 가장 큰 문제점은 동적 타이핑 규칙이 명확하지 않았던 것이다. 기존 마이크로소프트 스몰베이직 환경에서 스몰베이직 언어 의미(Semantics)에 관한 문서가 없을 뿐 아니라, 오픈 소스가 아니기 때문에 이러한 동적 타이핑 규칙을 파악하기 어려웠다. 이 연구에서 리버스 엔지니어링을 통해 동적 타이핑 규칙을 분석하여 다음 설명과 같이 정리하였다. 이러한 규칙을 명확히 정의하는 것은 스몰베이직 개발환경을 구현할 때 매우 중요하다.

스몰베이직의 동적타이핑 규칙은 다음과 같이 설명할 수 있다. 스몰베이직의 타입은 궁극적으로 문자열 타입 하나만 존재하지만, 문자열 타입 내부에서 각 타입에 맞게 변환해 주는 과정이 존재한다. 예를 들어,

- 식 “123” + 456은 문자열 “123”을 숫자 123으로 먼저 변환하고 덧셈하여 579를 계산하고,
- 식 “123” + “abc”는 두 문자열을 합하여 “123abc”로 계산하고,
- 문장 If “123” Then stmt1 Else stmt2는 문자열 “123”을 논리값 False로 먼저 변환하고 stmt2를 실행한다.
- 첫 번째 문장 arr = “123”을 실행하여 변수 arr에 문자열 “123”을 넣고, 식 arr[0]을 계산하면 문자열 “123”을 원소가 없는 빈 배열로 변환한 다음 첨자 0의 원소를 읽으려고 시도한다.

위의 예와 같이 문자열을 숫자, 논리값, 배열로 변환하는 방법을 해석기에서 지원해야 한다. 그리고 다른 자료형도 마찬가지로 동적으로 변환하는 방법이 필요하다.



(그림 4) Class Hierarchy for Dynamic Typing in Small Basic

[그림 4]과 같이 해석기에서 사용하는 모든 값은 Value 클래스 또는 자식 클래스(Number, String, Boolean, Array) 객체로 표현한다. 기반 클래스 Value에서 getNumber(), getString(), getBoolean(), getArray(), isNumber(), isBoolean(), isArray()의 추상메소드를 두고, 각 자식 클래스에서 이 메소드를 오버라이딩하는 방식으로 앞에서 설명한 동적 타이핑 변환을 구현한다.

앞에 스몰베이직 식과 문장의 예제를 설계한 클래스를 이용하여 다음과 같이 해석기의 동적타이핑이 필요한 부분을 구현할 수 있다. 예를 들어, 식 “123” + 456과 “123” + “abc”을 실행할 때 필요한 + 연산자를 Number와 변환가능한지 확인하여 숫자 더하기와 문자열 붙이기로 각각 다음

과 같이 구현할 수 있다.

```
Value op1 = ...;
Value op2 = ...;
Value r;
If (op1.isNumber() && op2.isNumber()) {
    r = new NumberV(
            op1.getNumber() + op2.getNumber());
}
else {
    r = new StrV(op1.getString() + op2.getString());
}
```

세 번째 예제 If “123” Then stmt1 Else stmt2에 대해 아래와 같이 조건식 “123”을 계산한 결과인 condV로부터 논리값을 가져와서 stmt1 또는 stmt2를 실행하도록 해석기를 구성한다.

```
Value condV = ...;
If (condV.getBoolean())
  // stmt1을 실행
else
  // stmt2를 실행
```

네 번째 예제 arr[0]에 대해 배열 이름 arr에 바인딩된 값을 환경(env)에서 가져온 다음 getArray()를 통해 배열인지 확인 후 배열이 아니면 동적 변환하여 배열을 만든다. 이 배열의 첨자 0의 원소를 가져오도록 해석기를 구성한다.

```
Value arrObj = env.get(“arr”);
Value r = arrObj.getArray().get(“0”);
```

스몰베이직의 동적 타이핑 규칙이 명확하게 정의되고 난 후, 예제 프로그램의 실행 결과가 일치하게 되었다. 큰 프로그램을 돌리며 발생하던 문제 또한 규칙이 정의된 후 대부분 해결할 수 있었다.

이 연구를 통해서 구현된 MySmallBasic은 마이크로소프트의 스몰베이직보다 속도가 상대적으로 느린 것을 확인했다. 그 이유는 두 가지로 파악할 수 있었다. 첫째, MySmallBasic에서 라이브러리를 호출해주는 방식으로 Java reflection API를 사용하기 때문이다. 이 API는 동적 클래스 로딩을 수행하여 속도가 느리고, 게다가 그래픽스 라이브러리를 사용하는 경우 해당 라이브러리를 빈번하게 사용하여 더욱 속도가 느려진다. 둘째, 해석기 방식을 사용하기 때문에 프로그램을 표현하는 AST(Abstract Syntax Tree)를 반복해서 해석하는 오버헤드로 느려진다.

Small Basic Code

```
TextWindow.WriteLine("Hello World!!")
```

해석기 방식

```
if(expr instanceof MethodCallExpr) {
    ...

    Class c = getClass("TextWindow");
    Method m = c.getMethod("WriteLine", ArrayList.class);
    m.invoke(null, new StrV("Hello World!!"));

    ...
}
```

컴파일 방식

```
TextWindow.WriteLine(new StrV("Hello World!!"));
```

(그림 5) The Difference between Interpreter and Compiler

현재 MySmallBasic을 인터프리터 방식에서 [그림 5] 와 같은 컴파일 방식으로 바꾸어 속도 저하를 줄이는 방식을 구현할 계획이다.

5. 결론 및 향후 연구

이 논문에서 Java로 스몰베이직 언어 해석기를 개발한 내용과 기본 라이브러리를 구현한 내용을 리포트했다. 스몰베이직은 처음 코딩을 접하는 사람에게 매우 유용한 간단한 프로그래밍 언어의 장점을 유지하면서, 큰 스몰베이직 프로그램을 실행시킬 수 있는 완성도 높은 스몰베이직 해석기를 구현했다.

향후 스몰베이직 언어의 속도 개선을 위한 논의사항들을 해결할 예정이다. 또한 초보자가 쉽게 접근할 수 있는 GUI 환경을 구축하여 교육용 프로그래밍 환경을 만들 수 있을 것이다.

감사의 글

오픈소스 스몰베이직 프로젝트에 김지용, 박세영, 정승완, 조문영, 조성모가 함께 참여하여 이 논문에서 언급한 연구 결과에 기여하였다.

참고문헌

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

[2] Vijaye Raji, Erik Meijer, Expert to Expert: The Basics of Small Basic, https://channel9.msdn.com/blogs/charies/expert-to-expert-the-basics-of-smallbasic, 2009.

[3] Philip Conrod, Lou Tylee, The Developer’s Reference Guide To Microsoft Small Basic, Kidware Software LLC, 2010.