컴파일러 개발환경 Edu-IDEC를 이용한 컴파일러 수업모형 개발

권정훈† · 박은경†† · 성우경††† · 김현주†††† · 배종민†††††

요 약

컴파일러와 언어구현 교과목은 컴퓨터과학 교과과정에서 오랫동안 중요한 주제로 인식되어 왔다. 그것은 컴파일러에 관한 지식이 프로그래밍 언어와 시스템을 이해하는 데에 중요한 역할을 할 뿐 아니라, 컴파일러 기술이 많은 응용 영역에서 활용될 수 있기 때문이다. 그러나 제한된 자원과 시간적 제약 때문에 컴파일러 기술을 효과적으로 전달하기 위해서는 많은 노력이 필요하다. 본 논문에서는 교육용 컴파일러 개발환경인 Edu-IDEC을 이용한 컴파일러 수업모형을 제시한다. Edu-IDEC는 로봇 플랫폼에서의 컴파일러 개발환경으로서, 컴파일러 제작도구, 레퍼런스 컴파일러, 구문트리 시각화도구, 목적언어 시각화도구, 로봇제어기, 그리고 로봇시뮬레이터 등의 기능이 있으며, 이클립스 플러그인 기반으로 동작한다. 그리고 제시된 모형을 실제 수업에 적용하여 그 평가결과를 제시한다.

주제어 : 컴파일러, 컴파일러교육, NXT, 프로그래밍환경, 이클립스 플러그인

Development of a Compiler Teaching Model Using the Compiler Developing Environment Edu-IDEC

Jung-Hoon Kwon† · Eun-Kyoung Park†† · Woo-Kyung Sung†††
· Hyun-Ju Kim†††† · Jong-Min Bae†††††

ABSTRACT

Compiler and language implementation courses have long been recognized as an important subject in Computer Science curricula. It is because not only the knowledge for a compiler plays important roles in understanding programming languages and systems but compiler technologies can be used in many applications. However it requires much effort to teach effectively it due to limited resources and time restriction. We present a compiler teaching model using Edu-IDEC which is a development environment of educational compilers. Edu-IDEC is a tool on the robot platform. It uses the Eclipse plug-ins and has functions like compiler developing tools, a reference compiler, visualization tool of syntax tree, visualization tool of object language, NXT robot controllers, and its simulator. We also present the evaluation results for our model by applying it to an actual class.

Keywords : Compiler, Compiler Education, NXT, Programming Environment, Eclipse plug-in

† 준 회 원: 경상대학교 컴퓨터과학과 대학원
†† 정 회 원: 경상대학교 컴퓨터과학과
††† 정 회 원: ㈜넥슨 신사업본부 신사업1실 연구원
†††† 정 회 원: 경남과학기술대학교 컴퓨터융합공학과 교수
††††† 정 회 원: 경상대학교 컴퓨터과학과 교수 (교신저자)
논문접수: 2013년 08월 20일, 심사완료: 2013년 10월 25일, 게재확정: 2013년 10월 31일

1. 서 론

컴파일러 교과목은 예전부터 항상 ACM (Association for Computing Machinery) 컴퓨팅 교과과정에서 핵심 주제로 정착된 과목이다[1]. 그리고 많은 교육자들이 학부에서 컴파일러 교육의 중요성을 인식해 왔다. 무엇보다 컴파일러를 만들기 위해서는 원시언어와 목적언어에 대한 지식이 충분해야 하기 때문에 컴파일러에 관한 지식이 프로그래밍 언어와 시스템을 이해하는 데에 중요한 역할을 한다[2][3]. 또한 컴파일러는 그 내용이 복잡하여 소프트웨어 공학에서 요구하는 설계기술을 활용해야 하며, 자료구조, 알고리즘, 프로그래밍언어, 계산이론 등의 기술을 활용하기 때문에 컴파일러 기술은 많은 응용 영역에서 유용하게 활용될 수 있다[2][3].

그런데 컴파일러 이론과 구현에 관한 높은 복잡도 때문에 주로 고학년의 교육과정에 배치되어 있으며, 다루어야 할 교육내용이 방대하여 한 학기에 필요한 내용을 다루기가 어렵고, 상업용 컴파일러는 너무 복잡하여 한 학기용 컴파일러 개발 프로젝트에 알맞은 컴파일러가 별로 없다[2]. 또한 컴파일러의 목적코드는 실제 기계가 아닌 시뮬레이터로 실행시키기 때문에 현실감이 떨어진다[1].

이에 따라 컴파일러 이론에 관한 기본개념을 더 잘 이해시킬 수 있는 도구를 개발하는 연구[4][5]와 함께, 교육용 컴파일러 개발 프로젝트를 보다 쉽게 수행할 수 있는 프로그래밍 환경[6][7]에 관한 연구가 필요하고, 컴파일러 개발 프로젝트를 수행할 수 있는 정도의 규모를 가진 교육용 컴파일러가 필요하다[1][2][3].

이에 우리는 실습 위주의 컴파일러 교과목 운영을 도와주는 컴파일러 개발환경 Edu-IDEC (Educational Integrated Development Environment for Compiler Development)을 개발하였는데, 본 논문에서는 Edu-IDEC를 이용한 컴파일러 수업모형을 제시하고, 이를 실제수업에 적용하여 그 평가결과를 제시한다. Edu-IDEC에는 레고(LEGO) 마인드스톰 NXT 로봇을 목적시스템으로 하는 레퍼런스 컴파일러, 컴파일러 제작도구, 목적언어 테스트 도구, 코드생성 시각화 도구 등의 기능이 포함되어 있는 통합 개발 환경이다.

이 도구를 활용하여, 컴파일러의 기본 개념을 이해시키고, 한 학기용 컴파일러 개발 프로젝트를 수행하는 데에 필요한 기본적인 컴파일러를 제공하여 학생들이 스스로 그 기능을 재정의 혹은 확대 설계하고 구현할 수 있도록 유도하며, 레고 마인드스톰 NXT 로봇을 목적시스템으로 활용하여 현실감을 높이고, 교육용 컴파일러 개발 프로젝트를 보다 쉽고 편리하게 수행할 수 있는 프로그래밍 환경을 활용하는 학습모형을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 살펴보고, 3장에서는 Edu-IDEC를 소개하고, 4장에서는 한 학기용 실습을 위한 레퍼런스 컴파일러에 대해서 설명한다. 5장에서는 Edu-IDEC를 이용한 수업모형을 제시하고, 6장에서는 본 논문에서 제시한 수업모형을 적용한 사례와 효과를 기술한다. 마지막으로 7장에서는 결론 및 향후 연구방향에 대해서 논한다.

2. 관련 연구

RobotStudio[4]와 Edu-IDEC[5]는 교육용 컴파일러 개발과제를 보다 쉽게 수행할 수 있는 프로그래밍 환경이다. RobotStudio는 이클립스(Eclipse) RCP(Rich Client Platform)로 구현된 Cricket 로봇 컨트롤러 기반의 교육용 컴파일러 개발환경이다. RobotStudio는 편집기, 파일관리, 텍스트 출력창 등을 제공하는 GUI 작업환경, 자원관리기, 레퍼런스 컴파일러, 그리고 Cricket 로봇 시뮬레이터로 구성되어 있다. NXT 로봇 환경에 비하여 학생들의 흥미를 쉽게 유도하기 어렵고, 컴파일러 이해를 돕는 지원기능이 미약한 편이다.

Edu-IDEC는 RobotStudio와 그 기본 개념은 비슷한데, 프로그램의 편집과정과 컴파일과정, 그리고 디버깅 과정을 하나의 환경에서 모두 지원해주는 강력한 개발 도구인데, 레고 마인드스톰 NXT 로봇을 통하여 목적언어를 실행하고, 실행결과를 확인할 수 있도록 테스트 도구를 제공하며, 원시언어와 목적언어의 관계를 시각적으로 표현하여 코드생성 과정의 이해를 돕는 코드생성 시각화도구를 제공한다.

그 외에도 컴파일러 이론에 관한 기본개념을

더 잘 이해시킬 수 있는 도구를 개발하는 연구로 Frances[4]와 MieruCompiler[5]가 있다. Frances는 코드 생성을 이해하는 것을 도와주는 도구이다. 특히 MieruCompiler는 모든 연관된 슬라이스에 대하여 소스코드, 추상구문트리, 어셈블리어, 심볼 테이블, 스택 내용, 컴파일러 코드를 마우스를 옮길 때 마다 밝게 보여줌으로서, 컴파일되는 과정과 중간 결과물을 가시적으로 볼 수 있어서, 복잡한 컴파일러 이론을 보다 쉽게 이해할 수 있는 도구이다.

본 논문에서는 Edu-IDEC를 활용한 컴파일러 교육 모델을 제시하고, 이를 실제 교육에 적용한 결과를 제시한다.

이와 관련된 연구로는 첫째, 로봇 기반으로 컴파일러를 교육하는 Chirp on Crickets[3]이 있다. Java/C 언어 기반의 Chirp언어와 Handy Crickets이라는 마이크로 컨트롤러를 이용한 컴파일러 교육용 개발환경을 이용한 6주간의 교육과정을 정의하였다. 이 모델의 목적은 고학년에 개설된 과목을 중간학년에서 개설하도록 하여 더 많은 학생이 수강토록 유도하는 것이다. 실제로는 RobotStudio[6]을 수업에 활용한 것으로 볼 수 있다. 6주간의 수업 스케줄을 제공하지만, 사용한 도구가 수업에 활용되는 방법이 명확하지 않고, 학습효과에 대한 평가가 없다.

Li Xu는 Chirp on Crickets과 유사한 환경에서 Chirp-Scribbler 언어를 정의하고 이에 대한 레퍼런스 컴파일러를 개발하여 교육에 활용한 결과를 제시하였다[1]. 이는 학생들이 새로운 언어를 학습하는 데에 따르는 시간적 제약이 있고, 학습효과에 대한 평가가 없으며, 지원 도구의 기능이 미약한 편이다.

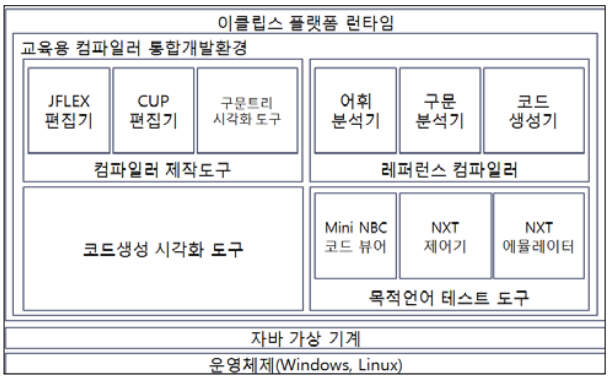
또 다른 연구로서 Bantam 프로젝트[2]는 한 학기에 컴파일러를 만들 수 있도록 자바 언어의 부분집합인 Bantam Java를 정의하고 이에 대한 컴파일러 개발을 통하여 컴파일러 과목을 학습하는 방법을 제시하였다. 대상 플랫폼을 MIPS와 X86으로 선정하여 교육용 컴파일러를 개발하였으며, 학생에게 컴파일러 개발 매뉴얼을 제공하여 컴파일러 학습에 활용하였다. 학습모형에 관한 내용이 미약하여 구체적인 교육효과에 대하여 알기 어렵다.

본 논문은 기존의 연구보다 강력한 교육지원 기능을 가지고 있는 Edu-IDEC를 활용한 컴파일러 수업모형을 제시하고, 이를 실제 교육에 적용한 결과를 제시한다.

3. Edu-IDEC 개요

Edu-IDEC은 교육용 컴파일러 통합 개발 환경으로 학생들이 컴파일러를 설계하고 제작할 수 있는 실습 환경이자 도구이다. Edu-IDEC은 프로그램의 편집과정과 컴파일과정, 그리고 디버깅 과정을 하나의 환경에서 모두 지원해 주는 강력한 개발 도구이다. Edu-IDEC은 다음의 특징을 가지고 있다.

- 이클립스 플러그인 프레임워크 기반으로 구현되었기 때문에 이클립스 통합 개발 환경이 동작하는 모든 플랫폼에서 사용이 가능하며, 실행 및 배포가 용이하다.
- 통합 개발 환경을 사용하는 학생의 접근성을 높일 수 있도록 GUI 기반으로 설계되었다.
- 학생들이 컴파일러를 설계하고 제작하는 과정에서 참고할 수 있도록 레퍼런스 컴파일러를 제공한다.
- 레고 마인드스톰 NXT 로봇을 통하여 목적언어를 실행하고, 실행결과를 확인할 수 있도록 테스트 도구를 제공한다.
- 원시언어와 목적언어의 관계를 시각적으로 표현하여 코드생성 과정의 이해를 돕는 코드생성 시각화도구를 제공한다.



<그림 1> Edu-IDEC 시스템 구조

<그림 1>은 Edu-IDEC의 시스템 구조이다. Edu-IDEC은 컴파일러 제작도구, 레퍼런스 컴파일러, 코드생성 시각화 도구, 목적언어 테스트 도

구로 구성되어 있다. 모든 모듈은 이클립스 플러그인 프레임워크 기반으로 구현되어 이클립스 플랫폼 런타임에서 수행된다.

3.1.1 컴파일러 제작도구

컴파일러 제작도구는 JFLEX 편집기, CUP 편집기, 구문트리 시각화 도구로 구성된다. JFLEX와 CUP은 Java 언어 기반의 어휘분석기, 구문분석기 생성 도구로서, 각각의 명세에 따라 Java 언어로 구현된 어휘분석기와 구문석기가 생성된다. JFLEX와 CUP은 이클립스 플러그인으로 제공되며 명세작성 및 생성을 이클립스 기반의 GUI 환경에서 수행한다. 그리고 구문트리 시각화 도구는 학생들에게 구문분석이 올바르게 수행되었는지 확인하는데 사용되고, 노드와 노드들의 관계를 박스와 화살표 형식으로 나타냄으로서 학생들에게 구문분석 과정의 이해를 돕게 한다.

3.1.2 레퍼런스 컴파일러

레퍼런스 컴파일러의 원시언어는 C언어 부분집합으로 정의되며 NXT 로봇 제어를 지원하는 함수를 포함한다. 목적언어는 NXT 로봇의 중간언어인 NBC[8]의 부분집합인 Mini NBC이다. 레퍼런스 컴파일러가 지원하는 원시언어와 목적언어는 학생들이 원시언어와 목적언어를 배우는 시간을 최소화하고, 컴파일러 소스코드를 단시간에 습득할 수 있도록 정의되었다.

3.1.3 코드생성 시각화 도구

코드생성 시각화 도구는 저급언어에 익숙하지 않는 학생들에게 코드생성 과정을 보다 쉽게 이해할 수 있도록 제공된 도구이다. 구문 트리 노드에서 생성되는 목적코드를 트리형태로 표시하여 구문트리와 목적언어의 관계를 시각적으로 표현하여줌으로서 코드생성 알고리즘을 쉽게 이해하는 데에 도움을 준다.

3.1.4 목적언어 테스트 도구

목적언어 테스트 도구는 목적언어인 Mini NBC를 실행하는 환경과 실행결과를 확인하는 이클립스 통합 개발 환경 기반의 프로그램이다. 이는 Mini NBC 코드 뷰어, NXT 제어기, NXT 에뮬레이터로 구성되어 있다. Mini NBC 코드 뷰어는 NBC 문법 구문강조 기능을 제공하여 학생들에게 목적언어의 구조를 보다 쉽게 이해할 수 있도록 한다. NXT 제어기는 목적언어를 레고 마인드스톰 NXT 로봇이 사용하는 NXT 바이트코드로 변환하고, 변환된 바이트코드를 NXT 로봇에 업로드하여 최종적으로 로봇을 실행하는 도구이다. NXT 에뮬레이터에서는 가상 NXT 로봇이 센서값을 제어할 수 있고, LCD 출력창과 실행속도를 제어할 수 있기 때문에 NXT 로봇이 없는 가상환경에서는 프로그램을 실행할 수 있도록 해준다.

4. 레퍼런스 컴파일러

레퍼런스 컴파일러는 컴파일러의 원리를 이해하고 구현을 돕는 가이드라인 역할을 한다. 아울러 실습중심의 컴파일러 수업에서 학습 범위에 대한 예를 제공한다.

4.1 원시언어

레퍼런스 컴파일러의 원시언어의 문법적 구조는 C 언어와 유사한 구조를 가지고 있다. <그림 2>는 원시언어의 문법을 BNF로 표현한 것이다.

```
<program>      ::=   <nxt_control>
<nxt_control>  ::=   <var_decl> void NAME ( ) <comp_stmt>
<var_decl>     ::=   <dcl_list> | ε
<dcl_list>     ::=   <dcl_list> <decl> | <decl>
<decl>         ::=   int <var_list> ;
<var_list>     ::=   <var_list> , NAME | NAME
<stmt>         ::=   <asgn_stmt> | <if_stmt> | <call_stmt>
                   | <while_stmt> | <comp_stmt>
<asgn_stmt>    ::=   NAME = <exp> ;
<if_stmt>      ::=   if ( <cond> ) <stmt> | if ( <cond> ) <stmt> else <stmt>
<while_stmt>   ::=   while ( <cond> ) <stmt>
<call_stmt>    ::=   NAME ( <param_list> ) ;
<param_list>   ::=   <exp_list> | ε
<exp_list>     ::=   <exp_list> , <exp> | <exp>
<comp_stmt>    ::=   { stmt_list}
<stmt_list>    ::=   <stmt_list> <stmt> | <stmt>
<cond>         ::=   <exp> >= <exp> | <exp> <= <exp>
                   | <exp> != <exp> | <exp> = <exp>
<exp>          ::=   <exp> + <term> | <exp> - <term> | <term>
<term>         ::=   <term> * <factor> | <term> / <factor> | <factor>
<factor>       ::=   NAME | NUMBER | HEXA | ( <exp> )
```

<그림 2> 원시언어의 문법

원시언어의 문법은 변수에 대한 선언부가 먼저 나오고 다음에 함수 선언 및 정의부가 나와야 한다. 그리고 함수 정의부에는 문장이 온다. 변수의 자료형은 정수형만 지원하며 함수는 사용자 정의

함수를 지원되지 않고 내장함수를 호출하여 사용한다. 함수 호출문은 함수명과 함수에 전달할 인자 값을 괄호에 포함한다. 인자 값으로는 변수, 상수, 문자열이 와야 한다. 변수명은 알파벳, 숫자, 또는 밑줄이 올 수 있으며, 첫 글자는 알파벳이 와야 한다. 문자열은 함수의 매개변수로만 사용가능하다.

4.2 목적언어

레퍼런스 컴파일러의 목적언어는 Mini NBC 이다, NBC(Next Byte Codes)는 NXT 로봇 프로그래밍 할 수 있는 저급언어이며, 어셈블리어와 유사한 문법형태를 가지고 있다. NBC 언어에는 로봇을 제어하기 위한 NBC API와 상수를 제공한다. 레퍼런스 컴파일러에서는 NBC 언어의 문법적 요소를 축약하여 교육목적에 부합하도록 Mini NBC 언어를 정의하였다. 이는 학생들에게 목적언어를 이해시키는 데에 최소한의 시간을 할당하기 위해서이다.

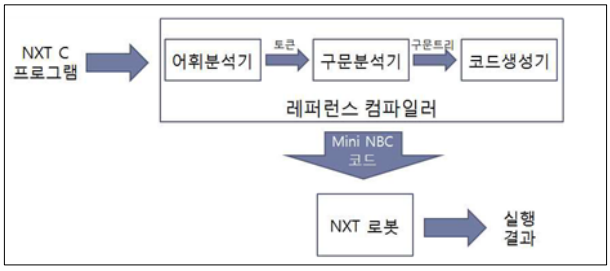
<표 1> NBC 명령어 일부

기능	Mini NBC 명령어
변수 선언 블록	dseg segment ~ dseg ends
실행문 블록	thread ~ endt
데이터 이동	mov, set
연산	add, sub, mul, div, mod, cmp
무조건 분기	jmp
조건 분기	`brtst

<표 1>은 Mini NBC 명령어의 일부이다. Mini NBC 언어는 변수 선언 블록과 실행문 블록으로 나뉜다. 변수 선언 블록은 변수 선언 및 정의가 가능하며 “dseg segment”로 시작하여 ”dseg ends”로 끝난다. Mini NBC 언어에서 제공하는 변수의 타입은 8bit unsigned 값을 저장할 수 있는 byte 형을 제공한다. 실행문 블록은 “thread main”으로 코드 수행을 시작하고 ”endt”를 끝으로 코드를 마친다. 그리고 실행문 블록에서 사용할 수 있는 명령은 이동 명령, 연산 명령, 무조건 분기 명령, 조건 분기 명령, 비교 명령 등이 있다.

4.3 레퍼런스 컴파일러 구조

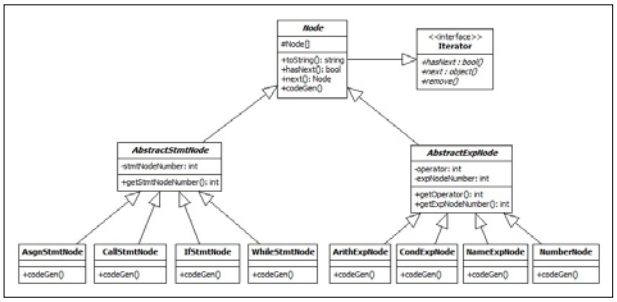
레퍼런스 컴파일러는 Java 언어를 기반으로 구현되어 있다. 레퍼런스 컴파일러는 Java 언어를 기반으로 하는 어휘분석기와 파서 생성 도구인 JFLEX와 CUP으로 구현된다. 레퍼런스 컴파일러의 구조는 <그림 3>과 같다.



<그림 3> 레퍼런스 컴파일러 구조

4.3.1 어휘분석기

어휘분석기는 자바 기반의 어휘분석기 생성기인 JFLEX[8]를 이용하여 구현된다. 어휘분석기는 사용자가 정의한 원시 프로그램을 입력받아 문자와 문자열을 토큰 단위로 구문분석기에 전달한다. 토큰에 대한 인식은 Lex와 마찬가지로 정규표현식을 이용한다. JFLEX의 명세를 통해 어휘분석기가 생성되는데, 생성된 어휘분석기는 Java 언어 기반의 어휘분석기가 생성 된다.

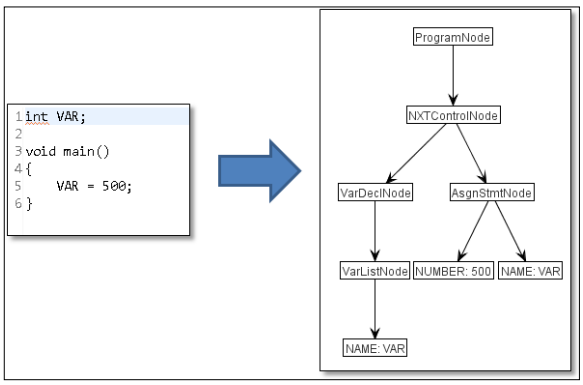


<그림 4> Node 클래스 다이어그램

4.3.2 구문분석기

구문분석기는 자바 기반의 파서생성도구인 CUP[9]을 이용해서 구현된다. 구문분석기는 어휘분석기가 제공하는 토큰을 입력으로 받아 구문트리를 생성한다. CUP의 명세에서 터미널(terminal)과 논터미널(nonterminal)을 정의하고 원시언어의 BNF를 정의한다. 그리고 BNF에서 문법에 대한 액션코드를 정의하여 구문분석기를 생성한다.

<그림 4>는 NXT C 언어의 구문트리 노드의 구조를 나타낸다. 생성된 구문트리의 노드 구조는 Node 추상 클래스와 Node 클래스를 상속한 클래스들로 구성된다. Node 추상 클래스를 상속받는 클래스에는 문장을 표현하는 AbstractStmtNode 추상 클래스, 수식을 나타내는 AbstractExpNode 추상 클래스로 구분된다. 배정문, 호출문, 제어문, 반복문은 AbstractStmtNode 추상 클래스를 상속받아 정의되고, 산술연산, 비교연산, 상수, 식별자는 AbstractExpNode 추상 클래스를 상속받아 정의된다. 그 외 노드는 Node 추상 클래스를 상속받아 정의된다. <그림 5>는 좌측의 코드에 대하여 생성된 구문트리의 예이다.



<그림 5> 인식한 구문에 대한 구문 트리

4.3.3 코드생성기

코드 생성기는 구문분석기에서 생성된 구문트리를 입력받아 Mini NBC 코드를 생성한다. 코드 생성과정은 각 노드 클래스에서 재정의(override)한 codeGen 메소드를 호출하면서 코드생성을 수행한다. codeGen 메소드에서는 해당 노드에서 생성해야 될 목적 언어의 코드를 생성하고, 자식 노드의 codeGen 메소드를 호출하면서 트리를 순회한다.

5. Edu-IDEC을 이용한 컴파일러 수업 모형 설계 및 적용

본 장에서는 Edu-IDEC 을 이용한 실습 프로젝트 중심의 컴파일러 수업 모형을 제시한다.

5.1 교육 과정

<표 2>는 Edu-IDEC을 기반으로 한 10주간의 컴파일러 교육과정안이다. 교육과정은 크게 Mini NBC, 어휘분석기, 구문분석기, 목적 코드 생성기 4부분으로 구성된다. Edu-IDEC은 이클립스 기반으로 구현되어 있다. 1주차는 Edu-IDEC을 이클립스 플러그인을 이용하여 실습환경을 구축하고, 목적 시스템인 레고 마인드스톰 NXT 로봇을 작동시키기 위한 USB 드라이버 및 라이브러리를 설치한다. 그리고 이클립스에서 Mini NBC 언어를 이용하여 레고 마인드스톰 NXT 로봇을 통해서 그 결과를 확인하기 위해 목적 언어 테스트 도구의 사용법을 학습한다.

2~3주차는 레퍼런스 컴파일러의 목적 언어인 Mini NBC 언어의 문법을 소개하며, 간단한 Mini NBC 언어를 이용한 프로그래밍 문제를 제공한다. 이때 Mini NBC 코드 뷰어를 통해서 디버깅을 편하게 하며, NXT 제어기를 활용하여, NBC를 사용한 로봇 프로그래밍한 결과를 로봇의 동작으로 확인한다.

4~6주차는 어휘분석기에 대해서 학습한다. 정규 표현식을 소개하고, 어휘분석기 제작도구인 JFLEX의 구조와 명세 방법을 익힌다. 이때 Edu-IDEC가 제공하는 JFLEX 편집기를 사용한다. 그리고 레퍼런스 컴파일러의 어휘분석기를 분석하고 그 기능을 확대한 어휘분석기 제작 과제를 수행한다.

7~9주차에서는 구문분석기 생성기인 CUP 사용법을 익히고 CUP 응용 프로그램을 작성하는 과제를 수행한다. 이때 Edu-IDEC의 CUP 편집기를 사용한다. 구문분석기의 출력결과는 구문트리이다. Edu-IDEC의 구문트리 시각화도구를 사용하여 생성된 구문트리의 구조를 확인하고, 구문트리에서 사용되는 Node 클래스를 통하여 트리노드의 구조를 분석하고, CUP을 활용한 구문분석기 소스코드를 분석한다.

마지막 10주차에는 구문트리를 순회하며 코드를 생성하는 알고리즘과 함께 코드 생성기의 소스코드를 분석한다. 이때 코드생성 소스코드를 이해를 돕기 위하여 구문트리 시각화도구를 활용하고, 생성된 코드가 올바른지 확인하기 위하여

NXT 제어기를 활용하여 NXT 로봇을 통해서 컴파일러에 오류가 있는지 확인한다.

<표 2> 교육용 컴파일러 통합개발환경을 이용한 학습 모형

주차	강의 주제	내용	활용도구
1	실습 환경 구축	• 마인드 스톰 NXT USB 드라이버 및 라이브러리 설치 • 이클립스 플러그인 설치 • NBC를 이용한 마인드 스톰 NXT 로봇 테스트 • 목적 시스템 소개	
2	Mini NBC	• NBC 언어의 구조 • 자료형 • 변수 선언 • 배정문 • 연산자 • 함수 정의 & 호출 • 목적언어의 기본적 문법	• Mini NBC 코드 뷰어, • NXT 제어기
3		• 분기문 • NBC API • NBC를 이용한 연습문제 • 목적언어의 문법 응용	• Mini NBC 코드 뷰어 • NXT 제어기
4	어휘 분석기	• JFLEX 문법 구조 • JFLEX 응용	• JFLEX 편집기 • 레퍼런스 컴파일러(어휘분석기)
5		• 토큰과 정규표현식 • JFLEX 옵션 • 어휘분석기 실습 • 토큰과 정규표현식	• JFLEX 편집기 • 레퍼런스 컴파일러(어휘분석기)
6		• 어휘분석기 설계 • 원시언어의 어휘분석기 소스코드 분석	• JFLEX 편집기 • 레퍼런스 컴파일러(어휘분석기)
7	구문 분석기	• CUP 문법 구조 • JFLEX와 CU의P 연동 • CUP 응용	• CUP 편집기 • 레퍼런스 컴파일러(구문분석기)
8		• 원시언어의 CFG • 구문 트리와 Node 구조	• CUP 편집기 • 레퍼런스 컴파일러(구문분석기)
9		• Node 클래스 구현 • 구문 분석기 소스코드 분석	• CUP 편집기, • 레퍼런스 컴파일러(구문분석기) • 구문트리 시각화 도구
10	코드 생성기	• 코드 생성 알고리즘 • 코드 생성기 소스코드 분석	• 구문트리 시각화 도구 • Mini NBC 코드 뷰어 • NXT 제어기 • NXT 에뮬레이터

5.2 교육과정 적용

<표 2>와 같은 내용으로 실제 수업에 적용한 결과에 대하여 논한다. 수강생은 기본적으로 Java 언어와 이클립스 환경에 대한 선수학습이 이미 이루어진 학생들로 제한하였다. Edu-IDEC을 이용하여 수업을 진행하면서 실제 학생들에게 레퍼런스 컴파일러를 참고하여 새로운 문법기능을 추가 혹은 보완하여 독자적인 컴파일러를 설계하고 제작하도록 하였다. 이에 대한 결과물로서 4개의 보고서(어휘분석기, 구문분석기, 코드생성기, 총괄)와 소스코드를 차례대로 제출토록 하였고, 이를 이용하여 학생들의 프로젝트를 평가하였다.

<그림 6>은 본 수업을 신청한 학생이 제출한 어휘분석기 보고서 중 JFLEX 명세의 일부이다. 레퍼런스 컴파일러에서 제공된 명세를 기반으로 새로운 어휘가 추가되었으며 이에 맞게 정규표현식 또한 새롭게 정의하였다.

```
                - 생략 -
CHAR = "'"[a-zA-Z]+"'"
NUMBER = [0-9]+.?[0-9]+
NAME             =       [_A-Za-z][_A-Za-z0-9]*
ARRAY = {NAME}"["[1-9][0-9]?"]"
newline                  =       \r\n|\n|";"
whitespace               =       [ \t]+
comment          =       "."+[A-Za-z0-9]+{newline}
THREAD = {NAME}"("")":"
%%
{comment}                { System.out.println("주석");}
"var"                    { System.out.print("variable "); }
"if"                     { System.out.print("if "); }
"other"                  { System.out.print("else if "); }
"for"                    { System.out.print("for "); }
":"                      { System.out.print("code block start "); }
","                      { System.out.print("Comma "); }
"equ"                    { System.out.print("equal "); }
"gt"                     { System.out.print("greate than" ); }
"ge"                     { System.out.print("greate equal " ); }
"lt"                     { System.out.print("less than" ); }
"le"                     { System.out.print("less equal" ); }
"ef"                     { System.out.print("code block end "); }
"eth"                    { System.out.print("thread end "); }
"="                      { System.out.print("assign "); }
"++"                     { System.out.print("PLUS+ "); }
"--"                     { System.out.print("MINUS- "); }
"+"                      { System.out.print("PLUS "); }
"-"                      { System.out.print("MINUS ");}
"*"                      { System.out.print("MUL ");}
"/"                      { System.out.print("DIV ");}
"%"                      { System.out.print("MOD "); }
"^"                      { System.out.print("Pow ");}
"out"                    { System.out.print("Print "); }
"in"                     { System.out.print("Input "); }
                - 생략 -
```

<그림 6> 어휘분석기 보고서 중 JFLEX 명세 일부

<그림 7>은 구문분석기 보고서 중 CUP에 대한 명세이다. 학생이 정의한 BNF의 stmt 논터미널의 생성규칙이 레퍼런스 컴파일러보다 훨씬 많고 다양한 기능이 제공되었으며, 다른 문법을 이용하여 제공되었음을 확인할 수 있다.

```
            - 생략 -
stmt        ::=     asgn_stmt:e1
                    {: RESULT = e1; :}
                    | if_stmt:e1
                    {: RESULT = e1; :}
                    | call_stmt:e1
                    {: RESULT = e1; :}
                    | while_stmt:e1
                    {: RESULT = e1; :}
                    | comp_stmt:e1
                    {: RESULT = e1; :}
                    | for_stmt:e1
                    {: RESULT = e1; :}
                    | self_stmt:e1
                    {: RESULT = e1; :}
                    | out_stmt:e1
                    {: RESULT = e1; :}
                    ;
self_stmt   ::=     NAME:e1 POSTPLUS SEMICOLON
                    {: RESULT = new SelfArithExpNode(SelfArithExpNode.PLUS, e1); :}
                    | NAME:e1 POSTMINUS SEMICOLON
                    {: RESULT = new SelfArithExpNode(SelfArithExpNode.MINUS, e1); :}
                    | NAME:e1 POSTPLUS
                    {: RESULT = new SelfArithExpNode(SelfArithExpNode.PLUS, e1); :}
                    | NAME:e1 POSTMINUS
                    {: RESULT = new SelfArithExpNode(SelfArithExpNode.MINUS, e1); :}
                    ;
asgn_stmt   ::=     NAME:e1 ASGN exp:e2 SEMICOLON
                    {: RESULT = new AsgnStmtNode(e1, e2); :}
                                ;
out_stmt    ::=     OUT factor:e1 COMMA NUMBER:e2 COMMA NUMBER:e3 SEMICOLON
                    {: RESULT = new OutStmtNode(e1, e2, e3); :}
                    |
                    OUT factor:e1 COMMA NUMBER:e2 COMMA NAME:e3 SEMICOLON
                    {: RESULT = new OutStmtNode(e1, e2, e3); :}
                    ;
for_stmt    ::=     FOR cond:e1 COMMA exp_list:e2 COLON stmt_list:e3 EF
                    {: RESULT = new ForStmtNode(e1, e2, e3); :}
                    | FOR cond:e1 COMMA self_stmt:e2 COLON stmt_list:e3 EF
                    {: RESULT = new ForStmtNode(e1, e2, e3); :}
                    ;
if_stmt     ::=     IF cond:e1 COLON stmt_list:e2 EF
                    {: RESULT = new IfStmtNode(e1, e2); :}
                    | IF cond:e1 COLON stmt_list:e2 OTHER COLON stmt_list:e3 EF
                    {: RESULT = new IfStmtNode(e1, e2, e3); :}
                    ;
while_stmt  ::=     WHILE PARAM_L cond:e1 PARAM_R stmt:e2
                    {: RESULT = new WhileStmtNode(e1, e2); :}
                    ;
call_stmt   ::=     NAME:e1 PARAM_L param_list:e2 PARAM_R SEMICOLON
                    {: RESULT = new CallStmtNode(e1, e2); :}
                    ;
            - 생략 -
```

<그림 7> 구문분석기 보고서 중
CUP 명세 일부

```
main():
        var i, j, tmp, arr[3], line, result;
        arr[1] = 1 * 3;
        line = 1;
        i = 1;
        j = 1;
        result = 1;
        arr[2] = 'It is !';
        arr[3] = 1.1;
        arr[3] = arr[3] + 1;
        arr[3] = 1 + arr[3];
        arr[2] = arr[2] + arr[3];
        arr[2] = arr[2] + ' Version !';

        .if tmp gteq 10:
        .       out 'not 10', 10, 1;
        .other:
        .       out '10', 10, 1;
        .ef

        for i le 10, i++:
                tmp = i % 2;

                if tmp equ 0 :
                        out i, 10, line;
                        out '*', 30, line;
                        out line, 50, line;
                        out '=', 70, line;
                        arr[1] = i * line;
                        out arr[1], 80, line;
                        line++;
                ef
        ef
        ClearScreen();
        out arr[2], 10, 1;
        out arr[3], 10, 2;
eth
```

<그림 8> 구문분석기 보고서 중
원시 코드의 예

<그림 8>은 구분분석기 보고서 중 원시코드의 예시이다. 배열을 추가하였고, 배열에는 문자열, 소수 등의 다양한 값이 저장되게 하였다. 또한 단항연산자, 비교연산자를 기호가 아닌 알파벳의 문자로 표시하는 등 다양한 문법적 기능을 제공하고 있음을 알 수 있다.

<그림 9>는 코드생성기 보고서 중 코드 생성의 결과인 NBC 코드의 예시 중 일부이다. 문자열을 할당받을 수 있는 변수의 자료형을 제공하기 위해 NBC 배열에 문자열을 받아서 처리하도록 코드를 생성하고 있다. 그리고 이외에도 레퍼런스 컴파일러에는 없는 다양한 기능을 정의하고 제공하고 있음을 확인할 수 있다.

```
dseg segment
  i byte
  j byte
  tmp byte
  arr1 byte
  arr2 byte
  arr3 byte
  line byte
  result byte
  tmp0 byte
                - 생략 -
  tmp11 byte
dseg ends

thread main
mul tmp10, 1, 3
mov arr1, tmp10
set line 1
set i 1
set j 1
set result 1
loop_begin19:
                - 생략 -
label19:
ClearScreen()
TextOut(10, LCD_LINE1, 'It is !3.1 Version !')
Wait(1000)
TextOut(10, LCD_LINE2, '3.1')
Wait(1000)
endt
```

<그림 9> 코드생성과제의 결과인
NBC 코드 일부

6. 평가

6.1 평가 대상 및 도구

본 논문에서 제시한 컴파일러 수업모형에 대한 효과를 측정하기 위해 컴퓨터과학과 4학년 '컴파일러 및 프로젝트' 교과목을 수강신청한 학생 15명을 대상으로 설문조사를 실시하였다. 평가도구는 참고문헌[11]에서 사용했던 도구를 본 연구에 맞게 수정하였는데, 이는 수업에 도구를 활용했을 때의 평가를 하는 공통점이 있기 때문이다. 구체적인 설문 문항의 내용은 <표 3>과 같다. 편리성,

적절성, 효과성, 만족도에 관한 문항에 덧붙여서 서술형 2문항을 추가하였다. 서술형을 제외한 각 문항은 '절대 아니다' 에 1점, '정말 그렇다'에 5점으로 구성된 5점 Likert 척도로 구성되어 있다.

<표 3> 설문 문항

평가 요소	평가문항
편리성	Edu-IDEC의 사용이 쉽고 편리하였다.
	Edu-IDEC을 활용하여 프로젝트를 수행하는데 어려움은 없었다.
	Edu-IDEC은 프로젝트를 수행하는데 필요한 환경 구축, 개발, 테스팅 등의 작업이 수월하였다.
적절성	프로젝트 진행과정은 적절하였다.
	Edu-IDEC는 프로젝트를 진행하는데 충분하게 기능을 제공하였다.
	개별 모듈을 활용하는 것에 비해 Edu-IDEC을 활용함으로써 프로젝트를 진행하는데 충분한 도움이 되었다.
효과성	Edu-IDEC을 이용하는 것이 프로젝트를 성공적으로 진행하는데 도움을 주었다.
	레퍼런스 컴파일러가 프로젝트를 진행하는데 충분한 예시가 되었다.
	Edu-IDEC이 컴파일러 교과목을 이해하는데 도움이 되었다.
만족도	본 학습에 열심히 참여하였다.
	본 학습에 흥미를 갖게 되었다.
	주변의 지인들에게 본 수업을 추천하고 싶다.
	컴파일러 수업의 좋았던 점과 개선해야 될 점.(서술)
	Edu-IDEC을 사용하면서 애로사항이나 개선해야 될 점.(서술)

6.2 평가 결과

<표 4>는 각 설문문항에 대한 평가 결과이다. 'Edu-IDEC이 컴파일러 교과목을 이해하는데 도움이 되었다.'의 문항이 평균 4.625로 가장 높은 점수를 받았으며, 이를 통해 컴파일러 교과목을 이해하는데 Edu-IDEC이 효과적이었음을 판단할 수 있다. 'Edu-IDEC을 활용하여 프로젝트를 수행하는데 어려움은 없었다.'의 문항이 평균 3.625로 가장 낮은 점수를 받았다. Edu-IDEC이 프로젝트를 수행하는데 도움은 되었지만 여전히 컴파일러 수업 자체에 대한 어려움을 가지고 있음을 알 수 있다. 그러나 전반적으로 Edu-IDEC이 컴파일러 수업을 이수하고 프로젝트를 진행함에 있어서 많은 도움이 되었으며, 학생들의 흥미 유발에도 긍정적인 효과를 보여주었다.

서술 문항에서는 실습시간이 조금 부족한 것 같아 아쉽다는 답변이 있었고, Edu-IDEC이 32bit 환경이라 실습을 진행하는 데에 어려움이 있었다는 답변이 있었다. 이 외에는 긍정적인 답변으로 Edu-IDEC을 통해서 큰 어려움 없이 프로젝트를 진행하고 수업을 이수할 수 있었다는 답변이 있었다.

<표 4> 설문 결과

내용	평균	표준편차
Edu-IDEC의 사용이 쉽고 편리하였다.	4.375	0.484123
Edu-IDEC을 활용하여 프로젝트를 수행하는데 어려움은 없었다.	3.625	0.695971
Edu-IDEC은 프로젝트를 수행하는데 필요한 환경 구축, 개발, 테스팅 등의 작업이 수월하였다.	4.375	0.695971
프로젝트 진행과정은 적절하였다.	4.5	0.5
Edu-IDEC는 프로젝트를 진행하는데 충분하게 기능을 제공하였다.	4.5	0.5
개별 모듈을 활용하는 것에 비해 Edu-IDEC을 활용함으로써 프로젝트를 진행하는데 충분한 도움이 되었다.	4.25	0.661438
Edu-IDEC을 이용하는 것이 프로젝트를 성공적으로 진행하는데 도움을 주었다.	4.375	0.484123
레퍼런스 컴파일러가 프로젝트를 진행하는데 충분한 예시가 되었다.	4.25	0.661438
Edu-IDEC이 컴파일러 교과목을 이해하는데 도움이 되었다.	4.625	0.484123
본 학습에 열심히 참여하였다.	4	0.707107
본 학습에 흥미를 갖게 되었다.	4.25	0.661438
주변의 지인들에게 본 수업을 추천하고 싶다.	4.25	0.433013

7. 결론 및 향후과제

본 논문에서는 Edu-IDEC을 이용한 컴파일러 수업 모형을 제시하고, 이를 실제 수업에 적용한 결과를 제시하였다. 본 논문에서 제시한 수업 모형의 궁극적인 목적은 Edu-IDEC을 이용하여 보다 쉽게 컴파일러를 직접 설계하고 제작함으로서 컴파일러 복잡도에 대한 어려움을 완화하고 학생들의 흥미와 동기 유발을 돕는 것이며, 나아가 컴파일러 구현을 통해서 다양한 응용영역에 대한 적응력을 키우기 위함이다.

학생들은 프로젝트를 수행하여 레퍼런스 컴파일러보다 많은 기능을 추가 혹은 보완하여 명세를 만들고 이에 대한 코드를 생성하였다. 그리고 본 모델을 수업에 적용한 결과, Edu-IDEC를 이용한 컴파일러 수업이 매우 효과적인 것으로 평가되었다.

Edu-IDEC은 이클립스 플러그인 프레임워크를

기반으로 구현되었기 때문에 확장이 가능하고 배포가 용이하다. 향후 컴파일러 이론을 더 잘 이해할 수 있는 다양한 도구가 개발되어 본 도구에 포함되면 더욱 효과적인 수업환경을 구축할 수 있을 것이다.

참고 문헌

[1] Li Xu (2008). Language Engineering in the context of Popular. Inexpensive Robot Platform. ACM SIGCSE'08, 43-48.

[2] M. L. Corliss, & E. C. Lewis (2008). Bantam: A Customizable, Java-Based, Classroom Compiler. ACM SIGCSE '08.

[3] Li Xu, & F. G. Martin (2006). Chirp on Crickets: Teaching Compilers Using an Embedded Robot Controller. ACM SIGCSE '06.

[4] T. Sondag, K. L. Pokorny, & H. Rajan (2010). Frances: A Tool For Understanding Code Generation. ACM SIGCSE'10.

[5] K. Gondow, N. Fukuyasu, & Y. Arahori (2010). MieruCompiler : Integrated Visualization Tool with Horizontal Slicing for Educational Compilers. ACM SIGCSE.10.

[6] Li Xu (2007). RobotStudio: A Modern IDE-based Approach to Reality Computing. SIGCSE '07.

[7] 성우경・강현석・배종민 (2011). 이클립스 기반 의 교육용 컴파일러 통합개발환경. 한국컴퓨터교육학회논문지, 14(5), 9-18.

[8] John Hansen. Next Byte Codes(NBC) Programmer's Guide Version 1.0.1 b33

[9] Gerwin Klein. JFlex - The Fast Scanner Generator for Java. http://jflex.de/

[10] Scott E. Hudson. CUP Parser Generator for Java, http://www.cs.princeton.edu/~appel/modern/java/CUP/

[11] 최봉선 (2008). 웹 기반 PBL에서 블로그 활용에 대한 연구. 한국정보교육학회논문지, 12(4), 385-393.

권 정 훈

2012 경상대학교
컴퓨터과학과(학사)
2013～현재 경상대학교
컴퓨터과학과 석사과정
관심분야: 컴파일러, 프로그래밍언어
E-Mail: boogierock27@gmail.com

박 은 경

1999 경상대학교
컴퓨터과학과(학사)
2002 경상대학교
컴퓨터과학과(석사)
2007 경상대학교 컴퓨터과학과 (박사)
관심분야: 컴퓨터교육, XML, DB 통합
E-Mail: pek1028@daum.net

성 우 경

2009 경상대학교
컴퓨터과학과(학사)
2011 경상대학교
컴퓨터과학과(석사)
2011～현재 (주)넥슨 연구원
관심분야: 프로그래밍언어, 컴파일러 최적화, 병렬 처리
E-Mail: sung_u_kyung@hotmail.com

김 현 주

1988 경상대학교
컴퓨터과학과(학사)
1990 숭실대학교
전자계산학과(석사)
2000 경상대학교 컴퓨터과학과 (박사)
2002～현재 경남과학기술대학교 컴퓨터융합공학과 교수
관심분야: 정보검색, 모바일 프로그래밍
E-Mail: khj@gntech.ac.kr

배 종 민

1980 서울대학교
수학교육과(학사)
1983 서울대학교
계산통계학과(석사)
1995 서울대학교 계산통계학과(박사)
1997～1998 Virginia Tech. 객원연구원
2011～2012 Univ. of Texas at El Paso 방문교수
1984～현재 경상대학교 컴퓨터과학과 교수
관심분야: 컴파일러, 프로그래밍언어, 컴퓨터교육
E-Mail: jmbae@gnu.ac.kr