프로젝트 #1

소프트웨어학부 컴파일러

2022년 9월 15일

문제

어휘분석 자동생성 도구인 Lex를 사용하여 COOL 컴파일러 구성 요소 중 첫 번째 단계인 어휘분석기 (lexical analyzer)를 구현한다.

COOL 문법

COOL의 문법은 아래와 같다. 여기서 굵은체는 키워드, 이탤릭체는 논터미널(non-terminal)을 의미한다.

program	::=	⟦*class*;⟧$^+$
class	::=	**class** TYPE [**inherits** TYPE] {⟦*feature*;⟧*}
feafure	::=	ID([*formal*⟦, *formal*⟧*]) : TYPE {*expr*}
	|	ID : TYPE [<- *expr*]
formal	::=	ID : TYPE
expr	::=	ID <- *expr*
	|	*expr*[@TYPE].ID([*expr*⟦, *expr*⟧*])
	|	ID([*expr*⟦, *expr*⟧*])
	|	**if** *expr* **then** *expr* **else** *expr* **fi**
	|	**while** *expr* **loop** *expr* **pool**
	|	{⟦*expr*;⟧$^+$}
	|	**let** ID : TYPE [<- *expr*] ⟦, ID : TYPE [<- *expr*]⟧* **in** *expr*
	|	**case** *expr* **of** ⟦ID : TYPE => *expr*;⟧$^+$ **esac**
	|	**new** TYPE
	|	**isvoid** *expr*
	|	*expr* + *expr*
	|	*expr* - *expr*
	|	*expr* * *expr*
	|	*expr* / *expr*
	|	~*expr*
	|	*expr* < *expr*
	|	*expr* <= *expr*
	|	*expr* = *expr*
	|	**not** *expr*
	|	(*expr*)
	|	ID
	|	integer
	|	string
	|	**true**
	|	**false**

구현 언어

이 프로젝트의 공식 프로그래밍 언어는 C이다. 프로젝트에서 제공하는 모든 골격 코드는 C언어로 되어 있다. C언어 이외에 개인이 선호하는 다른 언어를 사용할 경우 프로젝트 진행에 어려움이 따를 수 있다.

하지만 본인이 이런 어려움을 감수한다면 다른 언어를 사용하여도 무방하다. 다만 프로그래밍 언어에 따라 사용할 수 있는 Lex의 종류와 용법이 조금씩 다르므로 주의한다.

Flex 설치

Flex는 C/C++ 환경에서 가장 많이 사용하는 어휘분석기 자동생성 도구이다. 프로젝트를 수행하기 전에 먼저 Flex를 설치한다.

- **Linux** 환경에서는 터미널을 열고 다음 명령어를 실행한다.

  ```
  % sudo apt update
  % sudo apt install flex
  ```

  설치후 `flex` 명령어를 사용하여 Flex 소스파일을 컴파일하면 어휘분석기 코드인 `lex.yy.c`가 자동으로 생성된다. 이 프로그램을 `gcc`로 컴파일하고 링크시 `-ll` 옵션을 사용하면 어휘분석기 실행파일이 생성된다.

  ```
  % flex cool.l
  % gcc -o cool_lexter lex.yy.c -ll
  ```

- **macOS** 환경에서는 기본적으로 내장되어 있어서 별도의 설치가 필요없다. Flex를 사용하여 어휘분석기를 생성하는 과정은 Linux와 같다.

macOS 링크

macOS에서 실행파일을 만들기 위해 라이브러리를 링크할 때 아래와 같은, 혹은 유사한 경고 메시지가 발생할 수 있다.

```
ld: warning: object file (/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr
/lib/libl.a(libyywrap.o)) was built for newer macOS version (12.3) than being
linked (12.0)
```

심각한 오류는 아니지만 긴 메시지가 부담스럽게 느껴질 수 있다. 이것은 SDK 버전이 일치하지 않아서 생기는 것으로 `gcc` 컴파일러 옵션에 요구하는 SDK 버전을 명시하면 된다.

```
% gcc -o cool_lexter lex.yy.c -ll -mmacosx-version-min=12.3
```

골격 파일

프로젝트를 수행하는데 필요한 `Makefile`, `cool.skeleton.l`, `cool.tab.h` 파일을 학생에게 제공한다. `cool.skeleton.l`은 Flex로 작성된 `cool.l`의 골격파일로 어휘분석의 규칙을 정의하는 곳이다. 학생들은 이 파일에 필요한 규칙을 넣음으로써 어휘분석기를 완성한다. 헤더파일인 `cool.tab.h`에는 COOL 언어의 토큰이 정의되어 있으며 수정하지 않는다. `Makefile`은 모든 소스파일을 컴파일하여 실행파일 `cool_lexter`를 생성한다.

검증

어휘분석기를 검증하는데 필요한 `sample.cl`, `sample.cl.out`, `chk_examples` 파일과 다수의 COOL 프로그램이 들어 있는 `examples\` 폴더도 골격파일과 함께 제공한다. 어휘분석기가 완성되면 올바르게 동작하는지 주어진 `sample.cl`을 실행해본다.

```
% ./cool_lexer sample.cl
```

이 샘플파일은 완성된 COOL 프로그램은 아니지만 어휘분석기의 오류를 빨리 찾기 위해서 작성되었다. 이 샘플파일의 실행결과는 `sample.cl.out`과 일치해야 한다. 만일 일치한다면 어휘분석기가 거의 완성된 수준이며 마지막으로 `examples\` 폴더에 있는 COOL 프로그램을 실행해본다. 각 프로그램에 대한 실행결과는 같은 폴더 안에 있는 `COOL파일명.out`과 일치해야 한다. `chk_examples`는 셸스크립트로 실행과 검증과정을 자동으로 수행한다.

```
% ./chk_examples
examples/arith.cl --> PASSED
examples/atoi.cl --> PASSED
  :
examples/sort_list.cl --> PASSED
```

제출물

컴파일 과정과 실행결과를 보여주는 화면캡쳐, 소스코드, 그리고 자신의 결과를 보여주는데 필요한 부가 설명이나 기타 자료를 스스로 판단하여 제출한다. 모든 자료에는 학번과 이름을 명시하고, 소스코드를 제외한 나머지 제출물은 PDF 형식이어야 한다. 여기에는 다음에 열거한 것을 반드시 포함시켜야 한다.

- 컴파일(`make`)과 실행(`chk_examples`) 과정을 보여주는 화면 캡처
- 실행 결과에 대한 설명, 소감, 문제점
- Flex 소스파일 (`cool.l`) 별도 제출
- 샘플 프로그램 실행 결과 (`sample.cl.txt`) 별도 제출

HK