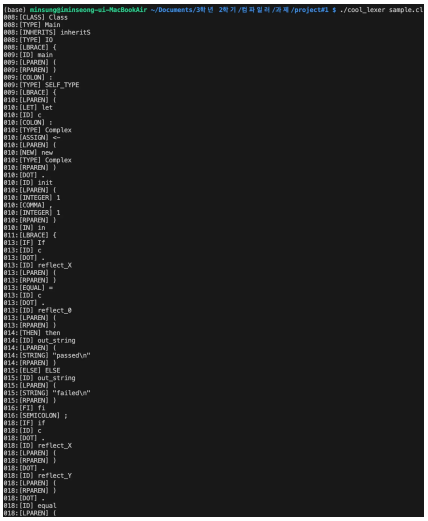
Compiler (CSE 4053) Project#1

응용물리학과 2022006971 이민성

컴파일 과정 :

```
(base) minsung@iminseong-ui-MacBookAir ~/Documents/3학년 2학기/컴파일러/과제/project#1 $ flex cool.l
(base) minsung@iminseong-ui-MacBookAir ~/Documents/3학년 2학기/컴파일러/과제/project#1 $ gcc -o cool_lexer lex.yy.c -ll
```

sample.cl 실행 과정 :



```
018:[ID] c
018:[DOT] .
018:[ID] reflect_0
018:[LPAREN] (
018:[RPAREN] )
018:[RPAREN] )
019:[THEN] then
019:[ID] out_string
019:[LPAREN] (
019:[STRING] "passed\n"
019:[RPAREN] )
020:[ELSE] else
020:[ID] out_string
020:[LPAREN] (
020:[STRING] "스트링 출력은 \"와 같은 탈출문자를 올바르게 처리해야 한다."
020:[RPAREN] )
021:[FI] fi
021:[SEMICOLON] ;
022:[RBRACE] }
023:[RPAREN] )
024:[RBRACE] }
024:[SEMICOLON] ;
025:[RBRACE] }
025:[SEMICOLON] ;
```

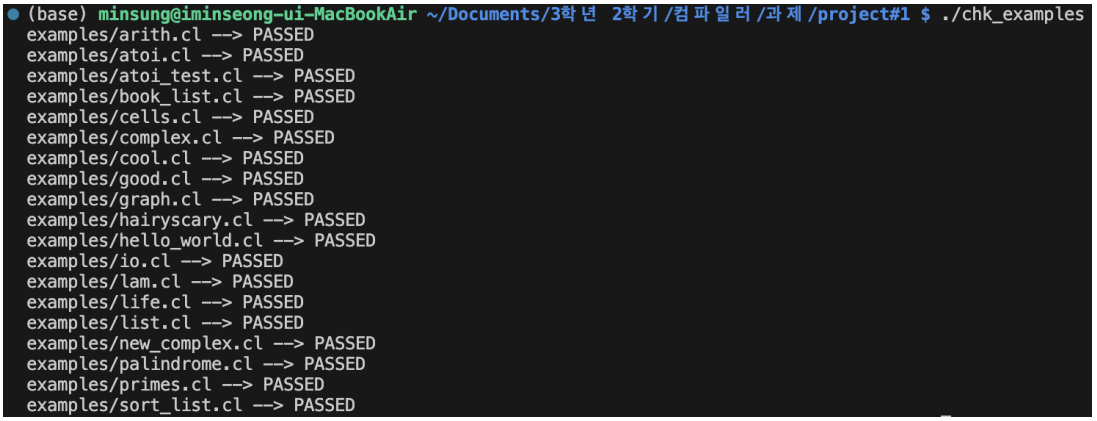
sample.cl.txt :

```
008:[CLASS] Class
008:[TYPE] Main
008:[INHERITS] inheritS
008:[TYPE] IO
```

```
008:[LBRACE] {
009:[ID] main
009:[LPAREN] (
009:[RPAREN] )
009:[COLON] :
009:[TYPE] SELF_TYPE
009:[LBRACE] {
010:[LPAREN] (
010:[LET] let
010:[ID] c
010:[COLON] :
010:[TYPE] Complex
010:[ASSIGN] <-
010:[LPAREN] (
010:[NEW] new
010:[TYPE] Complex
010:[RPAREN] )
010:[DOT] .
010:[ID] init
010:[LPAREN] (
010:[INTEGER] 1
010:[COMMA] ,
010:[INTEGER] 1
010:[RPAREN] )
010:[IN] in
011:[LBRACE] {
013:[IF] If
013:[ID] c
013:[DOT] .
013:[ID] reflect_X
013:[LPAREN] (
013:[RPAREN] )
013:[EQUAL] =
013:[ID] c
013:[DOT] .
013:[ID] reflect_0
013:[LPAREN] (
013:[RPAREN] )
014:[THEN] then
014:[ID] out_string
014:[LPAREN] (
014:[STRING] "passed\n"
014:[RPAREN] )
015:[ELSE] ELSE
015:[ID] out_string
015:[LPAREN] (
015:[STRING] "failed\n"
015:[RPAREN] )
```

```
016:[FI] fi
016:[SEMICOLON] ;
018:[IF] if
018:[ID] c
018:[DOT] .
018:[ID] reflect_X
018:[LPAREN] (
018:[RPAREN] )
018:[DOT] .
018:[ID] reflect_Y
018:[LPAREN] (
018:[RPAREN] )
018:[DOT] .
018:[ID] equal
018:[LPAREN] (
018:[ID] c
018:[DOT] .
018:[ID] reflect_0
018:[LPAREN] (
018:[RPAREN] )
018:[RPAREN] )
019:[THEN] then
019:[ID] out_string
019:[LPAREN] (
019:[STRING] "passed\n"
019:[RPAREN] )
020:[ELSE] else
020:[ID] out_string
020:[LPAREN] (
020:[STRING] "스트링 출력은 \"와 같은 탈출문자를 올바르게 처리해야 한다."
020:[RPAREN] )
021:[FI] fi
021:[SEMICOLON] ;
022:[RBRACE] }
023:[RPAREN] )
024:[RBRACE] }
024:[SEMICOLON] ;
025:[RBRACE] }
025:[SEMICOLON] ;
```

chk_examples 실행 과정 :



실행 결과에 대한 설명 :

cool.cl은 COOL 언어의 소스 파일을 분석하여 토큰을 식별하고 출력하는 어휘 분석기이다. 실행된 결과는 각 줄 번호, 토큰의 타입, 그리고 소스 코드에서 발견된 해당 토큰의 값을 출력하게 된다. 아래는 이 코드를 실행했을 때의 주요 동작 과정에 대한 설명이다.

- 주요 동작 과정 :
  1. 주석 처리 :
     - `"(*"`으로 시작하는 블록 주석은 중첩될 수 있으며, `comment_depth`를 사용해 주석의 깊이를 추적한다. 주석이 닫히는 `"*)"`를 만나면 깊이를 줄이고, `comment_depth`가 0이 되면 주석이 끝났다고 판단하여 분석을 계속한다.
     - `--`로 시작하는 한 줄 주석은 줄 끝까지 무시된다.
  2. 예약어 및 키워드 처리 :
     - `class`, `if`, `else`, `then`, `fi`와 같은 COOL 언어의 예약어를 우선적으로 처리한다.
     - 예약어는 대소문자를 구분하지 않도록 처리되어 있다. 예를 들어, `If`와 `if`는 모두 `IF`로 반환된다.
  3. 타입 처리 :
     - 타입은 대문자로 시작하는 식별자로 처리된다. 예를 들어, `Main`, `SELF_TYPE`, `Complex` 같은 식별자는 타입으로 간주되어 `TYPE`으로 반환된다.
     - Flex에서는 패턴 `[A-Z][a-zA-Z0-9_]*`로 정의되어 대문자로 시작하는 모든 단어가 타입으로 인식된다.
  4. 식별자 처리 :
     - 알파벳 또는 밑줄(`_`)로 시작하는 문자열은 식별자로 처리되며 `ID`로 반환된다. 예약어와 타입에 해당하지 않는 문자열이 식별자로 간주된다.
  5. 숫자 처리 :
     - 연속된 숫자는 정수로 인식되어 `INTEGER`로 반환된다.
  6. 문자열 처리 :
     - 큰따옴표(`"`)로 감싸인 텍스트는 문자열로 처리되어 `STRING`으로 반환된다. 문자열 내에 줄바꿈은 허용되지 않으며, 이스케이프 문자를 처리한다.
  7. 기호 및 연산자 처리 :

- 괄호, 중괄호, 산술 연산자, 비교 연산자, 할당 연산자 등은 각 기호에 대응하는 토큰(`LPAREN`, `RPAREN`, `PLUS`, `ASSIGN` 등)으로 반환된다.

8. 에러 처리 :
   - 인식할 수 없는 문자가 등장하면, 에러 메시지를 출력하고 프로그램을 종료한다.

실행 결과에 대한 소감 :

해당 Flex 코드의 실행 결과는 전반적으로 COOL 언어의 어휘 분석기로서 기대한 바를 충실히 수행하고 있다. 토큰의 식별과 구분이 정확하게 이루어지고 있으며, 대소문자 구분 없이 예약어와 타입을 인식하고, 식별자와 연산자 또한 올바르게 처리되는 모습이 눈에 띈다.

1. 예약어 인식:
   - 코드에서 `class`, `inherits`, `if`, `then`, `else` 등의 예약어가 제대로 `CLASS`, `INHERITS`, `IF`, `THEN`, `ELSE` 등으로 반환되고 있다. 이는 예약어에 대한 Flex 규칙이 잘 정의되어 있다는 것을 의미하며, 예약어 처리 순서가 올바르게 설정된 것으로 보인다.
2. 타입 인식:
   - `Main`, `SELF_TYPE`, `Int`와 같은 타입이 `TYPE`으로 정확하게 인식되고 있다. 특히 대문자로 시작하는 식별자를 타입으로 처리하는 규칙이 잘 적용된 것이 인상적이다. 이는 COOL 언어의 특성에 맞게 타입과 식별자를 구분하는 데 성공한 부분이다.
3. 식별자 처리:
   - `main`, `x`, `out_string` 같은 일반 식별자는 `ID`로 정확히 처리되고 있다. Flex의 식별자 규칙이 예약어 및 타입과 충돌하지 않도록 잘 정의된 것이 잘 반영되고 있으며, 이로 인해 코드가 깔끔하게 실행되고 있는 점이 만족스럽다.
4. 정수 및 문자열 처리:
   - `5`, `10` 등의 숫자가 `INTEGER`로 정확히 인식되고, `passed\n`, `failed\n`, `스트링 출력은...` 같은 문자열이 `STRING`으로 반환되는 부분 역시 매우 자연스럽고 기대한 바와 일치한다. 문자열에 대한 이스케이프 문자 처리도 문제가 없어 보인다.
5. **기호 및 연산자 처리**:
   - 괄호, 중괄호, 할당 연산자(`<-`), 산술 연산자(`+`, `-`), 비교 연산자(`<=`, `=`) 등이 모두 정확히 인식되며, Flex 패턴이 올바르게 작동하고 있는 것을 확인할 수 있다.
6. **주석 처리**:
   - 코드에 포함된 블록 주석과 한 줄 주석이 모두 무시되며 어휘 분석에 영향을 미치지 않는 점도 매우 잘 구현되어 있다. 주석이 코드 해석에 방해되지 않고, 주석 내부에서 줄바꿈이 있어도 정상적으로 처리되는 모습이 좋다.

처음에 과제를 봤을 때는 엄청 길고 어려워 보여 굉장히 막막하다고 생각했지만 수업시간에 교수님께 배운 cool 언어와 flex를 적용해가며, 어휘 분석에 대한 개념을 생각하다보니 보다 쉽게 풀 수 있었던 것 같다.

실행 결과에 대한 문제점 :

sample.cl을 실행한 것이 sample.cl.out의 결과와 똑같고 chk_examples를 실행 한 결과가 모두 PASSED가 된 것을 보아 문제점은 없다고 생각한다.