컴파일러개론 Assignment 3 (hw03)

최종 문서 수정 날짜 : 2019-10-02

과제 출제 날짜 : 2019 년 10 월 2 일

과제 마감 기한 : 2019 년 10 월 15 일 23 시 59 분 59 초, 이러닝사이트 제출 (Delay 없음)

과제 제출 파일 : 소스코드 (MiniCPrintListener.java)및 보고서(.pdf)을 압축한 zip 파일

(**코드 인코딩 꼭 확인**, UTF-8 안할 시 주석 깨지므로 주석 점수 0)

과제내용

"ANTLR"를 이용하여 주어진 문법에 대해 mini C 파일을 pretty print 하는 프로그램을 작성하시오

- 블록이나 nesting 되어 들어갈 때는 4 칸 들여쓰되 **'.'을 찍음**
- If 등의 특수 절이나 함수 시작은 괄호를 함수 다음 줄에 표시한다.
- 2 진 연산자와 피연산 사이에는 빈칸을 1 칸 둔다. 예) x+y ➔ `x + y`
- 전위 연산자와 피연산자 사이에는 빈칸을 두지 않는다. 예) ++x
- 일반 괄호는 expression 에 붙여 적는다. 예) `(x + y)`

간단한 예 1)

입력 (test1.c)

```
if (x>0    ) { x=x   +1;}
```

출력

```
if (x > 0)
{
....x = x + 1;
}
```

// 들여쓰기 할 때 공백이 아닌 • 을 사용

간단한 예 2)

입력 (test2.c)

```
int max=500; void main(){
  int i; int j; int k;
  int rem; int sum;

  i = 2;
  while(i <= max) {
        sum = 0;
        k = i / 2;
        j = i;
        while ( j <= k) {
              rem = i % j;
              if (rem == 0) {
                    sum = sum + j;
              ++j;}
        }
        if  (i == sum) write(i);
        ++i;
  }
}
```

출력

```
int max = 500;
void main()
{
....int i;
....int j;
....int k;
....int rem;
....int sum;
....i = 2;
....while (i <= max)
....{
........sum = 0;
........k = i / 2;
........j = i;
........while (j <= k)
........{
............rem = i % j;
............if (rem == 0)
............{
................sum = sum + j;
................++j;
............}
........}
........if (i == sum)
........{
............write(i);
........}
........++i;
....}
}
```

주의사항

- 문법(g4 파일)은 함께 게시되는 것을 사용한다.
- 반드시 Listener 를 사용 (다음 페이지에 상세설명)
- 그 외 궁금한 사항은 이러닝 질문답변 게시판에 질문할 것 (문법, 메인 등)

참고 사항

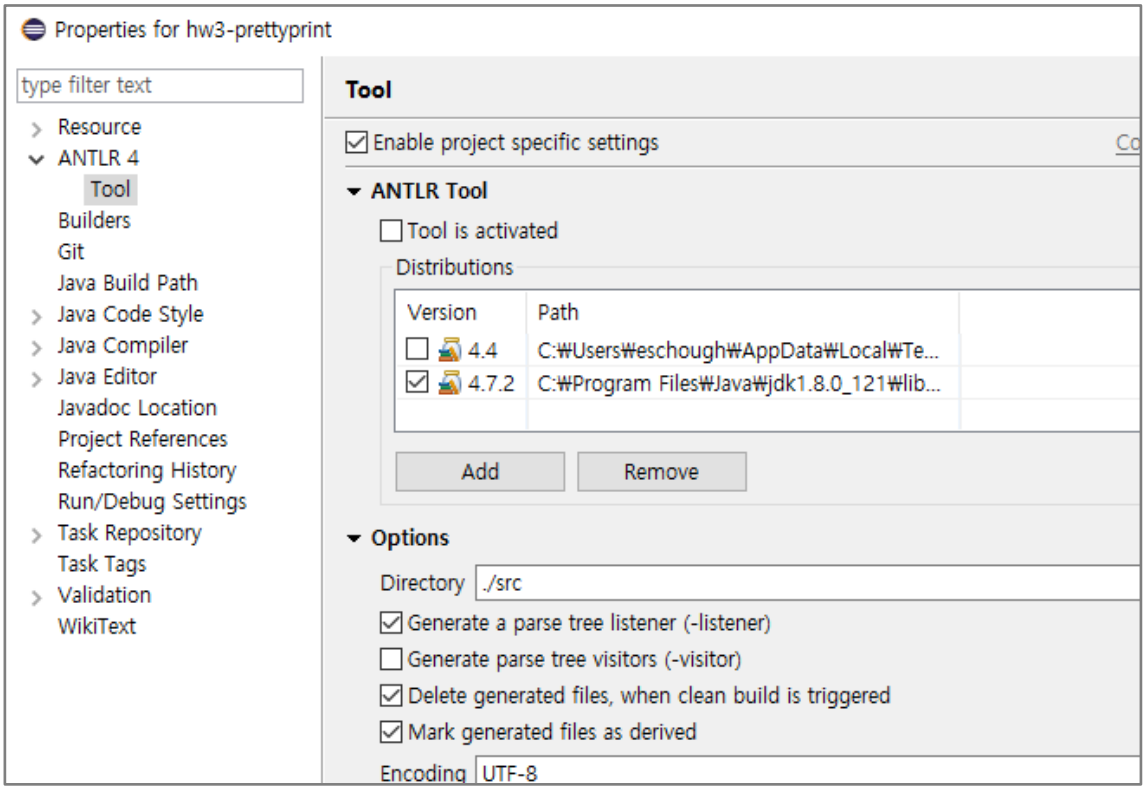
1. 문법 예 (MiniC.g4)

```
// 문법 이름이 MiniC 이다.
grammar MiniC;

// 나중에 ANTLR4 가 어휘분석, 구문분석을 하는 .java 파일들을 생성할 때,
//  generated 라는 package 에 생성하게 된다.
@header {
  package generated;
}

// program 이 시작 심벌이다.
program       : decl+               ;
decl          : var_decl
              | fun_decl            ;
var_decl      :  type_spec IDENT ';'
.....
// 문법만 있고 코드는 없다 (주의! 과제 2 와 달라진 것)
```

2. Project-Properties-ANTLR-Tool-Option 을 보면 결과 생성 경로 (Directory) 가 있다. 이것은 ./src/ 로 맞추면 편하다.



3. 과제 메인 코드 (간단 버전)

```
package listener.main;        // 원하면 변경 가능
import org.antlr.v4.runtime.*;
import org.antlr.v4.runtime.tree.ParseTree;
import generated.*;

public static void main(String[] args) throws Exception
{
        CharStream codeCharStream = CharStreams.fromFileName("test");
        MiniCLexer lexer = new MiniCLexer(codeCharStream);
        CommonTokenStream tokens = new CommonTokenStream( lexer );
        MiniCParser parser = new MiniCParser( tokens );
        ParseTree tree = parser.program();

        MiniCPrintWalker walker = new ParseTreeWalker();
        walker.walk(new MiniCPrintListener(), tree);
}
```

4 MiniCPrintListener 클래스

```
import generated.*;
public class MiniCPrintListener extends MiniCBaseListener {
        @Override
        public void enterFun_decl(MiniCParser.Fun_declContext ctx) {

        }
}
```

클래스 MiniCPrintListener 는, ANTLR 에서 자동 생성한 클래스 MiniCBaseListener 를 계승하는 하위클래스로 만든다. 해당클래스에서 MiniCBaseListner 에 존재하는 함수를 상속받아 기능을 구현한다.

5. ANTLR Listener

- 인터페이스 MiniCListener 는 ANTLR 가 자동으로 생성한 interface 이다. 각 non-terminal 에 대해 enter, exit 메소드를 선언하고 있다.

  예) non-terminal 인 program 에 대해 enterProgram(), exitProgram() 메소드, non-terminal 인 var_decl 에 대해 enterVar_decl(), exitVar_decl() 메소드 등

- 클래스 MiniCBaseListener 는 ANTLR 가 자동으로 생성한 클래스이다. 각 non-terminal 에 대해 enterXXX, exitXXX 메소드의 skeleton 을 만들어준다.

- ANTLR 의 walker 가 walk 을 하면 파스트리의 각 노드를 depth-first 로 방문하는데, preorder 로 수행하고 싶은 것은 enterXXX 에, postorder 로 수행하고 싶은 것은 exitXXX 에 들어있게 된다.

- 프로그래머는 MiniCBaseListener 를 상속받아 새로운 서브클래스에서 원하는 메소드를 재정의해서 사용한다.

참고 1. 한 Context 는 Tree 의 한 노드이기도 하다.

https://www.antlr.org/api/Java/ 사이트 참조

참고 2. isBinaryOperation 과 같은 보조 메소드를 많이 만들어두고 활용하면 편하고, 코드도 읽기 쉽다.

참고 3. g4 문법 규칙의 각 부분의 의미와 해당 부분에서 실제로 어떤 토큰들이 매치되어 어떤 트리가 그려지게 될지 확실히 이해한 후 진행하는 것이 코딩에 여러모로 유리하다.

참고 4. 과제에서는 자식노드에서는 put()으로 자신의 ctx 에 새로운 text 를 저장하고 부모노드에서는 children 의 ctx'에 대해 get(ctx')로 읽어올 수 있다.

예)

```
public class MiniCPrintListener extends MiniCBaseListener {
    ParseTreeProperty<String> newTexts = new ParseTreeProperty<String>();

    boolean isBinaryOperation(MiniCParser.ExprContext ctx) {
        return ctx.getChildCount() == 3 &&
                ctx.getChild(1) != ctx.expr();
    }
    @Override
    public void exitExpr(MiniCParser.ExprContext ctx) {
        String s1  = null, s2 = null, op = null;

        if (isBinaryOperation(ctx)) {
                // 예: expr '+' expr
                s1 = newTexts.get(ctx.expr(0));
                s2 = newTexts.get(ctx.expr(1));
                op = ctx.getChild(1).getText();
                newTexts.put(ctx, s1 + " " + op + " " + s2);
        }
```

- 파스트리 중 expr : expr '+' expr 로 만들어진 노드가 있다면, ctx.expr(0), ctx.expr(1)이 자식노드의 ctx 들이 된다.

참고 5. 부모노드에 값을 전달하는 방법은 크게 두가지가 있다. (1) 원래 recursive decent parser 의 리턴 값으로 전달하는 것이 전형적인 방법이었다. (강의 중에는 Topdown 후반부에

나옴) 이것은 Visitor 패턴으로 구현될 수 있고, XML 의 DOM 파서와 유사하다. (2) 하지만 지금 Listener 방식은 XML 의 SAX 파서와 유사한 방식인데, 좀더 간단하다.

- ANTLR4 는 Listener/Walker 방식과 Visitor 방식 둘다 지원한다. Eclipse 에서는 프로젝트의 Project-Properties-ANTLR-Tool-Option 에서 체크박스로 선택할 수 있게 지원된다. (eclipse 기준). 잘 안되는 사람은 Package Explorer 에서 g4 파일을 우클릭한 후 RunAs-External Tools Configurations... - ANTLR 에서 Arguments 로 조정한다.

- **Listener 로 하고 싶으면 –listener –no-visitor** ➔ 이번 과제!
- Visitor 로 하고 싶으면 –no-listener –visitor
- 둘다 생성하고 싶으면 –listner –visitor