Programming Project #3: 수식언어의 중간코드 생성기 개발

[문제]

LR(1) Parsing과 Syntax Directed Translation을 이용하여 간단한 수식 언어(Expression)의 중간코드 생성기를 개발한다.

1. 수식언어

수식은 다음 요소들을 포함한다. (과제2에 나온 요소들은 과제2와 규칙이 동일함)

1) operators
   - binary operator: '+', '-', '*', '/', '='(assignment)
   - unary operator: '+', '-'
   - 괄호 ( )
   - operators의 우선순위는 C언어에서의 연산자 우선순위를 따른다.

2) constants
   - integer
   - double (지수형 제외)

3) variables
   - 영문자로 시작하고 영문자와 숫자가 반복적으로 나타날 수 있음
   - 앞 10자리로만 구분함
   - integer, double 값을 저장할 수 있음

4) numeric expressions
   - int op int => int
   - int op double or double op int => double
   - double op double => double

5) type declaration 및 type checking
   - 변수는 type을 선언하여야 한다(주의사항: 단계별로 다르게 적용함).
   - 변수의 선언시 초기화는 하지 못한다.
   - translation 단계에서 type을 체크하여 혼합식이면 항목 4에 따라 type 변환이 필요하다.
     e.g. int a; double x, y, sum;

6) statements
   - type declaration statement(항목 5 참조)와 assignment statement가 있다.
   - statement는 세미콜론(;)으로 끝난다.

e.g. sum = a + b; x = y = 0.0;

8) arrays

- array는 C언어와 같이 선언하고 사용한다.

  e.g. int a[2][3];
  b = a[1][0];

2. 중간코드

중간코드의 형식은 3-address code의 형식을 따른다. 구체적인 형식은 강의노트를 참조한다.

3. 입력과 출력

입출력 파일은 모두 텍스트 파일이다.

- 입력 - 수식 문장들로 구성된 프로그램 파일 (exp.in)
- 출력 – 중간코드 파일 (ic.out) 및 심볼 테이블 (stb.out)

- 수식 프로그램 파일(exp.in) 예시

```
int r;
double pi;
r = 5;
pi = 3.14;
area = pi * r * r;
cir = pi * r + pi * r;
```

- 중간코드 파일(ic.out) 예시

```
r = 5
pi = 3.14
t1 = inttoreal r
t2 = pi * t1
area = t2 * t2
t3 = inttoreal r
t4 = pi * t3
t5 = inttoreal r
t6 = pi * t5
cir = t4 + t6
```

- 심볼테이블 파일(sbt.out) 예시

```
name    type     offset    // 이 라인은 실제 저장되는 내용이 아님(설명하기 위해 추가)
r       int       0
pi      double    4
```

[구현 조건]

- Lexical analyzer는 lex(혹은 flex)를 이용하고 parser generator는 yacc(혹은 bison)을 이용한다.
- LALR(1) 방식을 이용한다.

[오류처리]

1. Lexical Error

어휘가 틀리면 "lexical error(라인번호:문자위치)"라고 출력하고 해당 문자를 출력

e.g. 3 + hello_3 - 3abc

=> lexical error(1:10): _ or lexical error(1:5): hello_3

lexical error(1:15): 3 or lexical error(15): 3abc

2. Syntax Error

Parsing 중 오류가 발견되면 "syntax error(라인번호)"를 출력

e.g. a = a 5;

=> syntax error(1)

3. Error Recovery

Parsing 중 하나의 오류가 발생해도 그 문장의 오류를 출력한 후 계속 parsing을 해나간다.

[단계별 구현]

단계별로 구현하여 일부라도 기능을 구현한 마지막 단계만 제출하면 된다. 제출하는 단계는 이전 단계의 기능을 모두 포함하도록 구현해야 한다. 완성되지 못한 기능은 보고서에 명확하게 밝힌다. Error 처리 및 error recovery는 각 단계에서 포함할 수도 하지 않을 수도 있다. 보고서에 error 처리 및 error recovery 구현 여부를 명확하게 밝혀야 한다.

단계 1 – 변수선언이 없는 수식

- 상수, 변수, 연산자, 지정문 등이 포함된 수식 프로그램
- type declaration, type checking, type conversion, array 등이 없음
- 중간 코드 명령에 혼합형 연산이 가능함
  e.g. t1 = 10 + 10.5
- symbol table에는 변수의 name만 저장됨

단계 2 – 변수선언이 포함된 수식

- 변수 선언이 있음
- type ckecking을 하고 필요시 type 변환을 해야 함
- symbol table에는 변수의 name, type, offset(변수의 상대적 주소)을 저장해야 한다.

단계 3 – array가 포함된 수식

- array type은 구조화되어 symbol table에 저장되고(강의노트 참조) 출력시에는 string형태로 출력한다.

[보고서 구조]

1. 서론
   - 과제 소개
   - 각 단계별로 구현된 부분과 구현되지 않은 부분을 명확하게 명시할 것.
2. 문제 분석
   - grammar rule 분석
   - Syntax Directed Definition, Syntax Directed Translation 등에 대한 설명
3. 설계
   - 주요 자료구조 (중간코드 형식, symbol table 구조 등)
   - lexical analyzer나 parser에서 사용하는 주요 module에 대한 설명
4. 수행 결과
   - 입력 파일
     단계별로 2-3개 씩의 프로그램을 만들 것
     syntax error 등 error가 있는 경우를 포함
   - 출력 파일

주의) 소스코드는 첨부하지 않아도 됨

[제출방법 및 제출일]

- "hw3_학번" 디렉토리를 만들고 그 아래 src 디렉토리를 만들어 소스코드(*.c, *.h, *.l, *.y), Makefile를 넣고 보고서(hw3.pdf)는 hw3_학번 디렉토리에 둔다. 단, flex나 bison 컴파일러를 실행하여 얻은 코드(lex.yy.c, *.tab.c, *.tab.h)는 포함시키지 않는다.
- **Makefile의 수행결과로 생성되는 실행 파일의 이름을 "parser_학번.out"으로 한다.**
- 아주Bb 과제게시판에 "hw3_학번" 디렉토리 전체를 압축하여 올릴 것. (파일명: hw3_학번.zip)
- 제출일: 2022년 12월 18일(일) 자정 (보고서 출력본 제출하지 않음)
- **주의: 이 과제는 마감기간 내에 제출해야 함. 지연 제출 없음.**