**درس کامپایلر**

**گزارش تمرین 1 - 14 اسفند 1402**

**نیوشا یقینی - 98522346**

**توضیحات گزارش :**

در ابتدای گزارش صورت سوال و کد اولیه آورده شده.

10 ایده در این گزارش بررسی شده، تا به جواب نهایی رسیده ایم، در هر ایده توضیحات تغییرات آن آورده شده، و حالات مختلف چک شده.

تغییرات کد هایلایت شده اند و نتیجه ی هر ایده زیر آن نوشته شده.

ایده ی شماره 10، جواب نهایی ست.

در زیر همه ی ایده ها، سوال هایی که برایم ایجاد شده بود آورده شده.

**صورت سوال:**

کد اولیه را به نحوی تغییر دهید که در حالات مختلف (جمع، تفریق، ضرب و تقسیم) بتوان هم از sign number ها استفاده کرد، و هم از unsigned number ها بتوان استفاده کرد.

2 مثال مهم زیر باید قابل اجرا باشند:

1. A=9+9
2. B=9++9

**کد اولیه:**

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr ADD term | expr SUB term | term;  
term: term MUL factor | term DIV factor | factor;  
factor: NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;

ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
NUMERICALVALUE: FLOAT | INTEGER;  
fragment FLOAT: [0-9]\*'.'[0-9]+;  
fragment INTEGER: [0-9]+;

**ایده 1:**

* اضافه کردن WS به expr
* تغییر factor، اضافه کردن ('+' | '-')\*

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr ADD term | expr SUB term | term | WS;  
term: term MUL factor | term DIV factor | factor;  
factor: ('+' | '-')\* NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;

ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
NUMERICALVALUE: FLOAT | INTEGER;  
fragment FLOAT: [0-9]\*'.'[0-9]+;  
fragment INTEGER: [0-9]+;

**نتیجه:**

* تولید درخت تجزیه برای هر 2 مثال درخت صحیح است.
* main به درستی کار نمیکند.

**ایده 2:**

* تعریف کردن متغیر جدید به نام SIGN در Lexer rules
* اضافه کردن SIGN به جای ('+' | '-')\*

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr ADD term | expr SUB term | term | WS;  
term: term MUL factor | term DIV factor | factor;  
factor: (SIGN)\* NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;

ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
NUMERICALVALUE: FLOAT | INTEGER;  
fragment FLOAT: [0-9]\*'.'[0-9]+;  
fragment INTEGER: [0-9]+;  
  
SIGN: '+' | '-';

**نتیجه:**

* درخت تجزیه برای مثال “B=9++9” کار نمیکند.

**ایده 3: (4 حالت)**

* بررسی کردن حالت های مختلف اضافه کردن SIGN در factor
* تغییر SIGN به حالت معمولی

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr ADD term | expr SUB term | term | WS;

term: term MUL factor | term DIV factor | factor;

1) factor: NUMERICALVALUE | SIGN? NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;

2) factor: NUMERICALVALUE | SIGN\* NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;

3) factor: NUMERICALVALUE | SIGN NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;

4) factor: NUMERICALVALUE | SIGN+ NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;

ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
NUMERICALVALUE: FLOAT | INTEGER;  
fragment FLOAT: [0-9]\*'.'[0-9]+;  
fragment INTEGER: [0-9]+;

SIGN: '+' | '-';

**نتیجه:**

در همه حالت ها:

* درخت تجزیه برای مثال “B=9++9” کار نمیکند.

**ایده 4: (3 حالت)**

* بررسی کردن حالت های مختلف اضافه کردن SIGN برای NUMERICALVALUE
* تبدیل SIGN به fragment

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr ADD term | expr SUB term | term | WS;

term: term MUL factor | term DIV factor | factor;  
factor: NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;

ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;

1. NUMERICALVALUE: SIGN? (FLOAT | INTEGER);
2. NUMERICALVALUE: SIGN? FLOAT | SIGN? INTEGER ;
3. NUMERICALVALUE: FLOAT | INTEGER | SIGN FLOAT | SIGN INTEGER;

fragment SIGN: '+' | '-';

fragment FLOAT: [0-9]\*'.'[0-9]+;  
fragment INTEGER: [0-9]+;

**نتیجه:**

در همه ی حالت ها:

* درخت تجزیه برای مثال “A=9+9” کار نمیکند.

**ایده 5: (2 حالت)**

* تغییر نحوه تعریف FLOAT و INTEGER

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr ADD term | expr SUB term | term | WS;  
term: term MUL factor | term DIV factor | factor;  
factor: NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;  
  
ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
NUMERICALVALUE: FLOAT | INTEGER;  
1) fragment INTEGER: [0-9]\*'.'[0-9]+ | SIGN [0-9]\*'.'[0-9]+;  
 fragment FLOAT: [0-9]\*'.'[0-9]+ | SIGN [0-9]\*'.'[0-9]+;  
  
2) fragment INTEGER: SIGN\* [0-9]+;  
 fragment FLOAT: SIGN\* [0-9]+;

1. fragment INTEGER: SIGN? [0-9]+;  
   fragment FLOAT: SIGN? [0-9]+;

fragment SIGN: '+' | '-';

**نتیجه:**

در حالت 1:

* درخت تجزیه برای مثال “B=9++9” کار نمیکند.

در حالت 2 و 3:

* درخت تجزیه برای مثال “A=9+9” کار نمیکند.

**ایده 6: (3 حالت)**

* اضافه کردن SIGN? و SIGN\* به term و expr (هر بار به جای SIGN? ، SIGN\* هم چک شده است.)
* برگرداندن SIGN به حالت معمولی

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;

1. expr: expr ADD SIGN? term | expr SUB SIGN? term | SIGN? term | WS;  
   term: term MUL factor | term DIV factor | factor;
2. expr: expr ADD term | expr SUB term | term | WS;  
   term: term MUL SIGN? factor | term DIV SIGN? factor | SIGN? factor;
3. expr: expr ADD SIGN? term | expr SUB SIGN? term | SIGN? term | WS;  
   term: term MUL SIGN? factor | term DIV SIGN? factor | SIGN? factor;

factor: NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;  
ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
NUMERICALVALUE: FLOAT | INTEGER;  
fragment FLOAT: [0-9]\*'.'[0-9]+;  
fragment INTEGER: [0-9]+;  
  
SIGN: '+' | '-';

**نتیجه:**

در همه حالت ها:

* درخت تجزیه برای مثال “B=9++9” کار نمیکند.

**ایده 7:**

* حذف ADD و SUB و اضافه کردن '+' و '-' به expr
* اضافه کردن حالت SIGN به factor

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr '+' term | expr '-' term | term | WS;  
term: term MUL factor | term DIV factor | factor;  
factor: SIGN? NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;  
  
*//ADD: '+';  
//SUB: '-';*MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
NUMERICALVALUE: FLOAT | INTEGER;  
fragment FLOAT: [0-9]\*'.'[0-9]+;  
fragment INTEGER: [0-9]+;  
  
SIGN: '+' | '-';

**نتیجه:**

* درخت تجزیه برای مثال “B=9++9” کار نمیکند.

**ایده 8: (3 حالت)**

* تعریف کردن متغیر SIGNNUMERICALVALUE
* و تبدیل SIGN به fragment

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr ADD term | expr SUB term | term | WS;  
term: term MUL factor | term DIV factor | factor;  
factor: NUMERICALVALUE | SIGNNUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;

ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
NUMERICALVALUE: FLOAT | INTEGER;  
fragment FLOAT: [0-9]\*'.'[0-9]+;  
fragment INTEGER: [0-9]+;  
  
1) SIGNNUMERICALVALUE: SIGN (FLOAT | INTEGER);  
2) SIGNNUMERICALVALUE: SIGN FLOAT | SIGN INTEGER;  
3) SIGNNUMERICALVALUE: SIGN? NUMERICALVALUE;

fragment SIGN: '+' | '-';

**نتیجه:**

در همه حالت ها:

* درخت تجزیه برای مثال “A=9+9” کار نمیکند.

**ایده 9: (2 حالت)**

* یه مرحله اضافه کردن به factor و تبدیلش به factornew و اضافه کردن SIGN به آن

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr ADD term | expr SUB term | term | WS;  
term: term MUL factor | term DIV factor | factor;

1. factor: SIGN? factornew;
2. factor: SIGN\* factornew;

factornew: NUMERICALVALUE | LPAREN expr RPAREN | STRING | ID;

ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
NUMERICALVALUE: FLOAT | INTEGER;  
fragment FLOAT: [0-9]\*'.'[0-9]+;  
fragment INTEGER: [0-9]+;  
  
SIGN: '+' | '-';

**نتیجه:**

* درخت تجزیه برای مثال “B=9++9” کار نمیکند.

**ایده 10:**

* تبدیل کردن number و signednum به syntax rule

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr ADD term | expr SUB term | term | WS;  
term: term MUL factor | term DIV factor | factor;  
factor: number | signednum | LPAREN expr RPAREN | STRING | ID;  
  
number: FLOAT | INTEGER;  
signednum: ('+' | '-')\* number;  
ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
FLOAT: [0-9]\*'.'[0-9]+;  
INTEGER: [0-9]+;

**نتیجه:**

* درخت تجزیه برای همه ی مثال ها کار میکند.
* main به درستی عمل میکند.

**بخش سوالات :**

1. چرا در حالت زیر وقتی SIGN رو از Lexer rule به حالت Syntax rule در می آوریم، مثال “A=9+9” کار نمیکند؟

grammar ArithmeticExpression;  
  
start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr ADD term | expr SUB term | term | WS;  
term: term MUL factor | term DIV factor | factor;  
factor: number | signednum | LPAREN expr RPAREN | STRING | ID;  
number: FLOAT | INTEGER;

signednum: (SIGN)\* number;

SIGN: '+' | '-';  
ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
FLOAT: [0-9]\*'.'[0-9]+;  
INTEGER: [0-9]+;

1. چرا در حالت زیر هنگامی که جای SIGN را عوض می کنیم و زیر ADD می آوریم مثال “A=9+9” کار میکند اما مثال “B=9++9” کار نمیکند؟

start: assigns EOF;  
assigns: assign NEWLINE\* assigns\* | assign;  
assign: ID '=' expr;  
expr: expr ADD term | expr SUB term | term | WS;  
term: term MUL factor | term DIV factor | factor;  
factor: number | signednum | LPAREN expr RPAREN | STRING | ID;  
  
number: FLOAT | INTEGER;  
*//signednum: ('+' | '-')\* number;*signednum: (SIGN)\* number;  
*//  
//signednum: (sign)\* number;  
//sign: ('+' | '-');  
//  
//SIGN: '+' | '-';*ADD: '+';  
SUB: '-';  
MUL: '\*';  
DIV: '/';  
LPAREN: '(';  
RPAREN: ')';  
STRING: '"'.\*?'"';  
WS: [ \t\r]+ -> skip;  
NEWLINE: '\n';  
ID: [a-z,A-Z] ([a-z,A-Z] | [0-9])\*;  
  
FLOAT: [0-9]\*'.'[0-9]+;  
INTEGER: [0-9]+;  
  
SIGN: '+' | '-';