REPORT

[Lab5]

과　　목 : 프로그래밍언어론01

담당교수 : 송수환 교수님

학　　과 : 컴퓨터공학과

학　　번 : 2021111971

이　　름 : 이재혁

dongguk UNIVERSITY

To Do - Parser

Decls params -> fun id **(params)**; 구현

```
// [Function]
private Decls params() {
    Decls params = new Decls();   // 함수의 매개변수는 여러개가 존재할 수 있으므로 Decls의 AST
    // TODO: [Implement the code of params]
     Type t = type();              // 매개변수의 타입을 저장하는 변수
     String id = match(Token.ID);   // 매개변수의 이름을 저장하는 변수
     params.add(new Decl(id, t));   // 함수의 매개변수에 현재 저장한 타입, 아이디 정보를 추가
     while (token == Token.COMMA) { // 하나의 매개변수를 저장했을 때 다음 토큰이 ',' 라면 다음 매개변수 존재
        match(Token.COMMA);         // 매개변수가 존재하는 동안 params에 해당 매개변수 정보 저장
        t = type();
        id = match(Token.ID);
        params.add(new Decl(id, t));
     }
    return params;
}
```

fun id 이후에 나오는 매개변수들의 정보를 저장합니다.

함수 내부에서 사용될 "변수들"이므로 Decls AST입니다.

',' 있는 동안 다음 매개변수가 존재하므로, 매개변수의 타입과 아이디를 파라미터 정보에 추가합니다.

Let letStmt() : let AST에 지역함수 선언 경우 추가

```
private Let letStmt () { // let문의 선언부에 내부에서 사용할 함수를 추가하는 경우 존재
// <letStmt> -> let <decls> <functions> in <stmts> end
    match(Token.LET);     // LET 토큰, 변수선언, 함수정의 순서로 토큰 확인
    Decls ds = decls();  // let문에서 사용될 변수의 선언

    // [Function]
    // TODO: [Implement the code for function declaration in let stmt]
    Functions fs = null; // let문에서 사용할 함수정보를 저장하는 변수
     if(token == Token.FUN) { // 지역변수 선언 다음 토큰이 FUN이라면 함수 정의 존재
        fs = functions();     // let문 내부에서 사용할 함수들 저장
     }

    match(Token.IN);
    Stmts ss = stmts();
    match(Token.END);
    match(Token.SEMICOLON);
    return new Let(ds, fs, ss);
}
```

let문에서 사용할 지역함수가 있다면 지역 변수 다음에 정의됩니다.

변수 선언 다음 토큰이 함수라면, 존재하는 함수 정보들을 fs에 추가합니다.

함수가 추가되지 않았다면 null값으로 AST에 저장되기 때문에, 지역함수가 없다는 것을 알 수 있습니다.

Call call(Identifier id) : id(arguments)일 때 호출 AST반환 구현

```
// [Function]
private Call call(Identifier id) { // 함수 호출 AST 반환
// <call> -> id(<expr>{,<expr>});
    // TODO: [Implement the code of call stmt]
    match(Token.LPAREN);                // id를 확인해 매개변수로 전달받은 후 함수 매개변수가 시작되는 소괄호 확인
    Call c = new Call(id, arguments()); // 전달받은 id와 arguments로 호출 인자 정보로 호출 AST생성
    match(Token.RPAREN);                // 호출인자를 모두 확인하고 종료 소괄호 확인
    match(Token.SEMICOLON);             // 문장 종료 확인
    return c;                           // 생성한 Call AST 반환
}
```

id 다음 소괄호가 존재한다면 (arguments)가 존재합니다. 함수의 호출로 id와 arguments(인자들)을 Call AST로 생성해 반환합니다.

To Do - Sint

State Eval(Call c, State state) -> 함수 호출 시 연산 구현

```
State Eval(Call c, State state) { // 함수 호출을 평가
    // TODO: [Fill the code of call stmt]
     Value v = state.get(c.fid); // 함수 호출에 해당하는 아이디로 state를 탐색해함수를 v에 저장
     Function f = v.funValue();  // v는 함수 호출로 저장된 값이므로 함수 인스턴스, 함수 AST를 f에 저장
     state = newFrame(state, c, f); // 함수호출시 내부연산을 위한 프레임 생성
     state = Eval(f.stmt, state);   // newFrame으로 추가된 state로 함수의 stmt를 평가
     state = deleteFrame(state, c, f); // 함수 호출 후 인자 정보들을 제거

    return state;
}
```

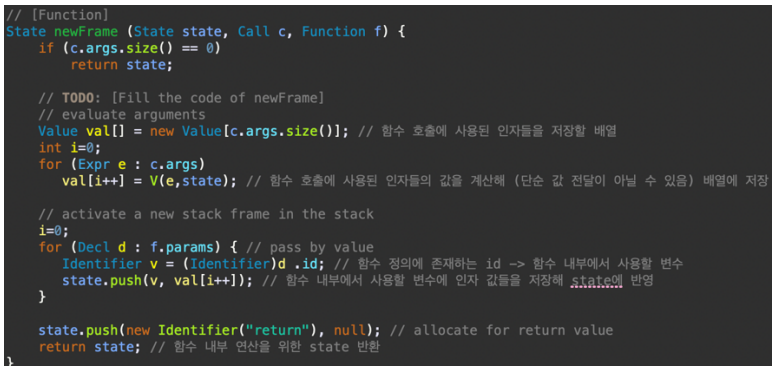
state에 저장된 함수를 불러와 함수 내부연산을 위해 state를 수정합니다.

수정한 state로 함수의 내부연산을 수행합니다.

함수가 종료되면 사용되었던 매개변수와 매칭된 인자들의 정보를 state에서 제거합니다.

State newFrame(State state, Call c, Function f)

-> 함수 내부연산에 사용할 변수 설정



```
// [Function]
State newFrame (State state, Call c, Function f) {
    if (c.args.size() == 0)
        return state;

    // TODO: [Fill the code of newFrame]
    // evaluate arguments
    Value val[] = new Value[c.args.size()]; // 함수 호출에 사용된 인자들을 저장할 배열
    int i=0;
    for (Expr e : c.args)
       val[i++] = V(e,state); // 함수 호출에 사용된 인자들의 값을 계산해 (단순 값 전달이 아닐 수 있음) 배열에 저장

    // activate a new stack frame in the stack
    i=0;
    for (Decl d : f.params) { // pass by value
       Identifier v = (Identifier)d .id; // 함수 정의에 존재하는 id -> 함수 내부에서 사용할 변수
       state.push(v, val[i++]); // 함수 내부에서 사용할 변수에 인자 값들을 저장해 state에 반영
    }

    state.push(new Identifier("return"), null); // allocate for return value
    return state; // 함수 내부 연산을 위한 state 반환
}
```

함수 내부 연산에 사용될 변수를 설정합니다. 변수의 id는 함수 정의 사용되었던 params이고, 그 값은 호출로 전달된 arguments들 입니다.

Value배열에 계산한 값들을 저장해 매개변수와 pair로 state에 저장됩니다.

리턴값을 저장할 변수를 state에 추가합니다.

State deleteFrame(State state, Call c, Function f)

-> 함수 연산 종료 후 매개변수 정보 제거

```
// [Function]
State deleteFrame (State state, Call c, Function f) { // 함수 연산 이후 관련 정보 제거
// free a stack frame from the stack
    state.pop();  // pop the return value

    // TODO: [Fill the code of deleteFrame]
    if (f.params != null)
       state  = free(f.params, state); // 함수 정의에 존재하는 매개변수 정보들을 state에서 제거

    return state; // 수정된 state 반환
}
```

함수 내부 연산 종료 후 해당 매개변수 정보들은 외부에서 더이상 사용하지 않기 때문에, state에서 매개변수들을 제거합니다.

State Eval(Let l, State state) -> let문에 존재하는 지역함수 정보추가

```
State Eval(Let l, State state) {
    State s = allocate(l.decls, state); // 지역변수로 사용할 변수들을 state에 할당

    // [Function]
    if (l.funs != null) { // let문에 지역함수 선언이 존재한다면
        // TODO: [Fill the code here]
        // Case with Function declaration inside Let
        for (Function fun : l.funs)
            s.push(fun.id, new Value(fun)); // 지역 state에 아이디와 해당함수를 pair로 저장
    }

    s = Eval(l.stmts, s); // 추가된 let지역 변수 함수로 stmt를 평가합니다.

    return free(l.decls, s); // let함수 종료 후 선언부를 state에서 제거합니다.
}
```

let문 선언부에 지역함수의 선언이 존재한다면 state에서 함수의 정보를 호출할 수 있도록 함수의 id와 함수 정보를 state에 추가합니다.

추가된 내용으로 let문 실행하고 지역변수를 state에서 제거합니다.

결과분석

hi8.s

```
Begin parsing... src/test/hi8.s

Interpreting...src/test/hi8.s

Interpreting...src/test/hi8.s
1

Interpreting...src/test/hi8.s

Interpreting...src/test/hi8.s
100
```

int 변수 x에 1을 저장합니다

x에 저장된 1을 출력합니다.

함수의 기능을 정의합니다. state에 f의 아이디와 기능이 추가됩니다.

함수 호출시 매개변수 x에 10의 인자 값이 pair를 이루어 state에 추가됩니다.

함수가 x*x의 값 10*10 즉 100을 출력합니다.

hi9.s

```
Begin parsing... src/test/hi9.s

Interpreting...src/test/hi9.s

Interpreting...src/test/hi9.s

Interpreting...src/test/hi9.s

Interpreting...src/test/hi9.s

Interpreting...src/test/hi9.s

Interpreting...src/test/hi9.s
120

Interpreting...src/test/hi9.s

Interpreting...src/test/hi9.s
true
```

int 변수 i를 0, bool 변수 b를 선언합니다.

g함수를 선언합니다. int 변수 2개를 매개변수로 받아 x, y로 사용합니다.

x > y면 true, 아니면 false를 반환합니다.

f함수를 선언합니다. int 변수 1개를 매개변수로 받아 x로 사용합니다.

x가 1이면 1, 그렇지 않으면 x * f(x-1)의 값을 반환합니다.

리턴에 f(x-1)이 호출되어 위 함수를 반복합니다 1을 리턴했을때 값이 계산됩니다.

처음 전달받은 x부터 1까지 곱하는 팩토리얼 함수입니다.

i에 f(5)의 리턴 값을 할당합니다. 120으로 계산되어 i에 저장됩니다.

f(id) 뒤에 '('로 함수 호출임을 알 수 있습니다.

i를 출력합니다.

b에 g(5, 3)의 리턴값을 할당합니다.

g(id) 뒤에 '('로 함수 호출임을 알 수 있습니다.

5 뒤에 ','로 인자가 더 존재한다고 판단합니다.

3 뒤에 ','가 없으므로 인자가 2개임을 알 수 있습니다.

b를 출력합니다.

hi10.s

```
Begin parsing... src/test/hi10.s

Interpreting...src/test/hi10.s

Interpreting...src/test/hi10.s

Interpreting...src/test/hi10.s

Interpreting...src/test/hi10.s
11
11
15
15
```

int 변수 i가 전역에 1로 선언되었습니다.

i에 매개변수 x를 더하는 f와 x*x를 더하는 g가 정의되었습니다.

let문이 선언되며 지역변수 i가 선언되었습니다.

지역 변수 i는 10이 할당 됩니다.

f(1)의 리턴값을 출력합니다. state에 지역 변수 i가 먼저 발견되기 때문에, 지역 변수 i에 1이 더해져 11을 저장합니다. i를 리턴해, 11이 출력됩니다.

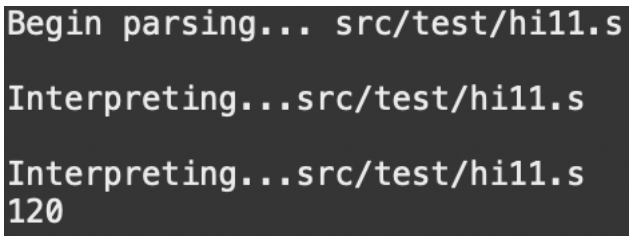
i를 출력합니다. 함수 f로 값이 11로 변했습니다.

g(2)의 리턴값을 출력합니다. 마찬가지로 i에 + 2*2가 더해져 15가 저장됩니다.

i를 리턴해 15를 출력합니다.

i를 출력합니다. 함수 g로 값이 15로 변했습니다.

hi11.s



int값을 반환하는 fact함수가 정의되었습니다. 매개변수로 전달받은 값을 n으로 사용합니다.

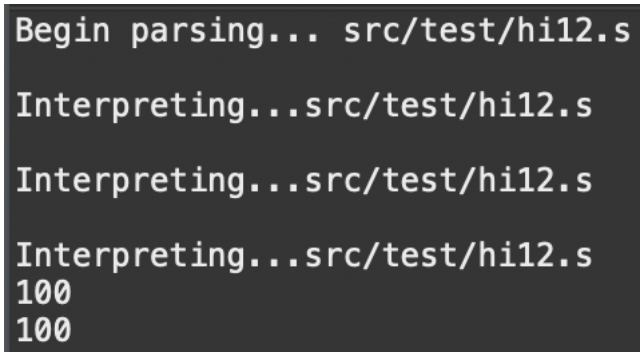
n이 1이면 1, 그렇지 않다면 n*fact(n-1)의 값을 리턴합니다.

fact(5)의 값을 출력합니다.

fact 함수의 매개변수 n에 5가 저장됩니다. 5 * f(4) 를 리턴할 때 f가 다시 호출됩니다. fact함수의 매개변수 n에 4가 저장됩니다. 위 과정을 반복해 n이 1일 때 1을 리턴해 5 * 4 * 3 * 2 * 1의 값이 리턴됩니다.

리턴된 120을 출력합니다.

hi12.s



int 매개변수 x, bool 매개변수 y를 사용하는 f함수를 정의합니다.

let문 으로 지역변수 result가 0으로 선언됩니다.

인자로 전달받은 y가 true면 전달받은 x의 제곱을 result에 저장하고 반환합니다.

int 매개변수 x를 전달받는 g함수를 정의합니다.

let문으로 지역변수 i와, true값으로 초기화 되는 b를 선언합니다.

g(10)을 호출해 리턴값을 반환합니다.

g의 x에 10이 저장됩니다. let문에서 지역변수 i, true값으로 초기화 되는 b가 선언됩니다.

f 함수에 x, b를 담아 호출하고 리턴 값을 i에 저장합니다.

f의 x에 10, y에 true가 저장됩니다. f의 let문에서 지역변수 result가 0의 값으로 선언됩니다.

y가 true 이므로, result에 10 * 10이 저장됩니다. result를 출력합니다. f에 의해 첫번째 100이 출력됩니다.

result값이 반환됩니다. state에서 f함수 관련 내용이 지워집니다.

g로 돌아와 i에 f(x,b)의 반환값인 100이 저장됩니다. i의 값을 반환합니다. . state에서 g함수 관련 내용이 지워집니다.

g(10)이 리턴한 100이 출력됩니다. 두번째 100이 출력됩니다.

hi13.s

```
Begin parsing... src/test/hi13.s

Interpreting...src/test/hi13.s
120
true
```

let문에 선언부에 int 변수 i가 0으로, bool 변수 b가 선언됩니다.

다음 토큰이 fun으로 함수정의가 있음을 알 수 있습니다.

함수 g가 정의됩니다. int 변수 2개를 전달받아 x, y로 사용합니다.

x > y라면 true, 아니라면 false를 리턴합니다.

함수 f가 정의됩니다. int 변수 1개를 전달받아 x로 사용합니다.

x가 1이라면 1을 리턴합니다. 그렇지 않다면 x * f(x-1)을 리턴합니다.

state에 해당 정보들이 저장됩니다. 변수들은 id와 값이, 함수들은 함수id와 기능이 저장됩니다.

지역변수 i에 지역함수 f(5)의 리턴값을 저장합니다. 120이 저장됩니다.

i의 값을 출력합니다. 120이 출력됩니다.

지역변수 b에 지역함수 g(5,3)의 리턴값을 저장합니다. true가 저장됩니다.

b의 값을 출력합니다. true가 출력됩니다.