REPORT

[Lab4]

과　　목 : 프로그래밍언어론01

담당교수 : 송수환 교수님

학　　과 : 컴퓨터공학과

학　　번 : 2021111971

이　　름 : 이재혁

dongguk UNIVERSITY

To Do - Parser

Decl decl() -> <type> id [n]; 구현

```
private Decl decl() {
   // <decl> -> <type> id [n];
   // <decl> -> <type> id [=<expr>];
   Type t = type();
   String id = match(Token.ID);
   Decl d = null;

   if (token == Token.LBRACKET) {
       // TODO: [Fill the code for array declaration (<type> id [n];)]
       // Use the AST of "Decl (String s, Type t, int n)"
       // ex) Value n = literal();, n.intValue() ...
       match(Token.LBRACKET);   // <type> id '['n] -> 대괄호 시작 확인
       Value n = literal();     // <type> id ['n'] -> n 내용 Value로 저장
       int size = n.intValue(); // n은 배열의 길이 정보이기 때문에 intValue이어야함 int타입의 size 변수에 저장
       d = new Decl(id,t,size); // 배열의 id, 타입, 크기정보로 AST생성
       match(Token.RBRACKET);   // <type> id [n']' -> 대괄호 종료 확인

   } else if (token == Token.ASSIGN) {
        match(Token.ASSIGN);
        Expr e = expr();
       d = new Decl(id, t, e);
   } else
        d = new Decl(id, t);

    match(Token.SEMICOLON);
    return d;
}
```

대괄호 안에 있는 n을 int로 계산해 size로 Decl객체(AST)를 생성합니다.
아이디, 타입, 크기 정보를 포함합니다.

크기 정보가 전달되어 배열의 선언임을 알 수 있습니다.

Stmt assignment() -> id[<expr1>] = <expr2>; 구현

```
private Stmt assignment() {
// <assignment> -> id[<expr>] = <expr>;
// <assignment> -> id = <expr>;

    Array ar = null;
    Identifier id = new Identifier(match(Token.ID));

    if (token == Token.LBRACKET) {  // id[<expr>] = <expr>;
        // TODO: [Fill the code for assignment to array elements]
        // Set the AST of Array(Identifier s, Expr e) to Array ar
        match(Token.LBRACKET);      // id'['<expr>] = <expr>; -> 대괄호 시작 확인
        Expr index = expr();        // id['<expr>'] = <expr>; -> 어느 index에 값을 대입하고 있는지 expr()로 저장
        ar = new Array(id, index); // id의 expr번째의 AST 생성
        match(Token.RBRACKET);      // id[<expr>']' = <expr>; -> 대괄호 종료 확인
    }

    match(Token.ASSIGN);            // id[<expr>] '=' <expr>; -> 할당 확인
    Expr e = expr();                // id[<expr>] = '<expr>'; -> 할당하는 Expr 저장
    match(Token.SEMICOLON);         // id[<expr>] = <expr>';' -> 세미콜론으로 문장 종료 확인

    if (ar == null)
        return new Assignment(id, e);
    else
        return new Assignment(ar, e); // ar정보가 생성되었다면 배열의 할당으로 AST 생성
}
```

id 배열의 <expr1>번째 index에 <expr2>의 값을 할당하는 Stmt입니다.

id의 index번째를 표현하는 Array객체(AST)를 생성합니다.

Expr factor() -> id'['<expr>']' 구현

```
    case ID:
        Identifier v = new Identifier(match(Token.ID));
        e = v;
        if (token == Token.LBRACKET) {  // id[<expr>]
            // TODO: [Fill the code for using array elements]
            // Set the AST of Array(Identifier s, Expr e) to Expr e
            match(Token.LBRACKET);     // id'['<expr>] -> 대괄호 시작 확인
            Expr index = expr();       // id['<expr>'] -> 어느 index에 값을 대입하고 있는지 expr()로 저장
            match(Token.RBRACKET);     // id[<expr>']' -> 대괄호 종료 확인

            e = new Array(v, index);  // 현재 factor는 v의 index위치의 값 표현
        }
        break;
```

id 배열의 <expr>번째 index의 값을 표현하는 factor 입니다.

'[' 다음에 오는 Expr은 접근할 배열의 index의 표현식이기 때문에 Expr객체로 저장해 Array객체로 묶습니다.

To Do - Sint

State Eval(Assginment a, State state) -> 배열 원소 할당 구현

```
State Eval(Assignment a, State state) {
    // replace array element in array represented by array name
    Value v = V(a.expr, state);

    if (a.ar == null)
        return state.set(a.id, v);
    else {  // id[<expr>] = <expr>;
        // TODO: [Fill the code for assignment of an array element]
        // Assign to the array element id[<expr>] of the array represented by the array name.
        // Replace array element in array represented by array name (not use state.set(id, value!)
        // ex) Value[] ar = ....;   // Array can be represented the array of Value
        //     ar[index] = v;       // Replace array element


        // Assignment에 배열(ar)이 존재한다는 것은 이미 state에 id에 해당하는 배열이 push되어 있는 상태
        // 즉 배열의 <expr> 인덱스에 v를 할당

        Value[] arr = state.get(a.ar.id).arrValue();
        // state에서 ar의 id (값을 바꿀 배열)을 탐색, array가 Value형태로 state에 존재
        // arrValue() 메소드로 Value[] 즉 배열의 형태로 arr 레퍼런스 선언

        arr[V(a.ar.expr,state).intValue()] = v;
        // 할당문에 배열이 있다는 것은 배열에 입력할 위치도 있음 (expr)
        // 배열의 어느 위치에 v를 삽입할지 계산 -> V(a.ar.expr,state)
        // V()는 Value 객체를 return하기 때문에 intValue() 메소드로 int값으로 반환
        // 계산한 index로 arr 레퍼런스 값 수정
        return state;
    }
}
```

매개변수 a객체에 배열의 id 할당할 index의 위치(Expr)가 저장되어 있습니다.
state에서 아이디를 찾아 Value 배열 레퍼런스로 지정합니다.

Expr을 int값으로 계산해 해당 위치에 v를 할당합니다.

State allocate(Decls ds, State state) -> 배열선언(state에 push) 구현

```
State allocate (Decls ds, State state) {
    // add entries for declared variables (ds) on the state
    if (ds != null) {
        for (Decl decl : ds)
            if (decl.arraysize > 0) { // <type> id[n];
                // TODO: [Fill the code for array allocation]
                // Regarding array declaration,
                // create an array (new Value[arraysize])
                // and push (id, new Value(array)) to the state.

                Value[] arr = new Value[decl.arraysize]; // decl에 저장된 배열의 크기만큼 Value 배열 레퍼런스를 선언
                state.push(decl.id, new Value(arr));     // 선언된 변수 id로 Value 배열 레퍼런스를 Value의 형태로 state에 저장
            }
            else if (decl.expr == null)
                state.push(decl.id, new Value(decl.type));
            else
                state.push(decl.id, V(decl.expr, state));
    }
    return state;
}
```

decl에 저장된 arraysize 크기의 Value 배열 레퍼런스를 생성합니다.

레퍼런스를 Value의 형태로 decl에 저장된 id와 함께 state에 push 합니다.

Value V(Expr e, State state) -> id[<expr>]의 값 반환 구현

```
Value V(Expr e, State state) {
    if (e instanceof Value)
        return (Value) e;

    if (e instanceof Identifier) {
        Identifier v = (Identifier) e;
        return (Value)(state.get(v));
    }
    if (e instanceof Array) { // id[<expr>]
        // TODO: [Fill the code for array expr]
        // For the array represented by the array name,
        // return the value of the array element id[<expr>]
        // ex) Value v = (Value) state.get(ar.id);
        //     Value[] vs = v.arrValue();
        //     return (vs[index])

        Array eArr = (Array)e; // e가 Array의 instance이므로, Array로 변환
        Value[] arr = ((Value)state.get(eArr.id)).arrValue(); // 배열은 Value 형태로 state에 저장, Array의 아이디로 찾은 Value를 배열로 변환
        int index = V(eArr.expr, state).intValue(); // Array객체에 <expr>로 어떤 index의 값에 접근하는지 저장되어 있음
                                                    // state에 저장된 정보로 expr를 계산후 intValue() 메소드로 index에 저장
        return arr[index]; // Value 배열의 index Value 객체를 리턴

    }
    if (e instanceof Binary) {
        Binary b = (Binary) e;
        Value v1 = V(b.expr1, state);
        Value v2 = V(b.expr2, state);
        return binaryOperation (b.op, v1, v2);
    }
    if (e instanceof Unary) {
        Unary u = (Unary) e;
        Value v = V(u.expr, state);
        return unaryOperation(u.op, v);
    }
    throw new IllegalArgumentException("no operation");
}
```

Array객체는 배열 id, Expr객체로 index를 저장하고 있습니다.

state에서 id를 탐색합니다. Value 배열은 Value의 형태로 저장되어 있어 arrValue() 메소드로 배열 형태로 반환해 레퍼런스를 선언합니다.

expr을 V()함수로 계산해 Value 객체로 변환 후 intValue() 메소드로 int 변수 index에 저장합니다.

해당 index에 해당하는 Value 객체를 반환합니다.

결과분석

q1.s

Begin parsing... src/test/q1.s Interpreting...src/test/q1.s Interpreting...src/test/q1.s Interpreting...src/test/q1.s Interpreting...src/test/q1.s Interpreting...src/test/q1.s 7	int a[3]; state에 a와, Value[3]을 멤버 변수로 가지는 Value객체가 Pair를 이루어 state에 저장합니다. a[1]=3; a[2]=7; a[3]=2; a와 Expr객체(index)를 멤버변수로 가지는 Array객체를 생성합니다. Array객체에 있는 a를 state에서 탐색하고 Value 배열을 찾습니다. Expr을 계산해 index를 얻고 배열의 index 위치에 할당 표현식을 Value로 계산해 저장합니다. print a[1]; a[1]에 해당하는 Array객체를 생성합니다. state에서 a를 찾아 1번 위치의 Value를 print합니다.

q2.s

```
Interpreting...src/test/q2.s
Sum
10
```

let의 decls에서 state에 a, Value(Value[5])가 저장됩니다.

while문을 반복하며 Identifier = a, Expr = i(반복하면서 변경됨)인 Array객체를 생성하고 state에서 a에 해당하는 Value[] 레퍼런스를 찾아 i의 위치에 i를 할당합니다. (a = [0, 1, 2, 3, 4])

while문을 반복하며 state에서 a에 해당하는 Value[] 레퍼런스를 찾아 i위치의 Value객체와 sum을 더해 sum에 할당합니다. (sum = sum + i | i : 0~4)

"Sum"과 sum에 저장된 값을 출력합니다.

q3.s

```
Interpreting...src/test/q3.s
Max
6
```

let의 decls에서 state에 a, Value(Value[5])가 저장됩니다.

Array 객체를 생성해 값 state에서 Value[]를 찾아 Value를 할당합니다.

(a = [1, 4, 3, 6, 2])

while문을 반복하며 a와 i로 Array객체를 생성하고 state에서 Value[]를 찾아 i번째 값을 max와 비교해 큰 값을 max에 저장합니다.

(max는 1, 3, 6)으로 변화합니다.

"Max"와 max에 저장된 값을 출력합니다.

q4.s

```
Interpreting...src/test/q4.s
Sum of positive numbers
22
```

let의 decls에서 state에 abc, Value(Value[8])가 저장됩니다.

while문을 반복하며 abc와 i로 Array객체를 생성합니다.

i*i - (i+3)*3 - 3; 의 값을 계산해,

state에서 abc에 해당하는 Value[]를 찾아 i번째에 저장합니다.

(abc = [-12, -14, -14, -12, -8, -2, 6, 16])

while문을 반복하며 abc와 i(i : 0~7)로 Array객체를 생성합니다.

state에서 abc를 찾아 i번째의 값이 0보다 크다면 sum에 더해 저장합니다.

"Sum of positivev numbers"와 sum의 값을 출력합니다.

q5.s

```
Interpreting...src/test/q5.s
A
U
F
G
A
B
E
true
```

string배열 ar1[6], int배열 ar2[6], bool배열 ar3[6]이 state에 저장됩니다.

Decl객체에 타입정보를 포함하고 있습니다.

ar1 = ["F", "A", "G", "U", "B", "E"]의 값으로 설정합니다.

while문으로

ar2 = [1, 2, 3, 4, 5, 6]

ar3 = [true, true, true, true, true, true] 의 값으로 설정합니다.

ar3[ar2[ar2[4]-4]] = ar3[2] = false;

ar2[1] = ar2[0] + ar2[1] = 1 + 2 = 3

ar2[2] = ar2[4] - ar2[ar2[ar2[1]]] = 5 - 5 = 0

ar2[3] = ar2[4]/(ar2[4] - ar2[1]) = 5/5-3 = 2

ar2[4] = ar2[1]*ar2[3] - (ar2[0]*2) = 3*2 - 1*2 = 4

ar2 = [1, 3, 0, 2, 4, 6]

while문으로

ar1[1], ar1[3], ar1[0], ar1[2], ar1[1](ar2[i] == 4, j = 4), ar1[4]를 출력합니다.

A U F G A B

ar1[j+1] = ar1[5]를 출력합니다.

E

ar3[3]을 출력합니다.

true