프로그래밍언어 개론 보고서

[09] Built in Function

제출일	2017/07/09
학 번	201000287
소 속	일어일문학과
이 름	유다훈

1 과제문제

1 Interpreter 만들기 (Built in Function 구현)

- 입력 String 이 기존에 선언한 문법에 따라 작동하게 하는 Interpreter 를 구현하기 위하여 키워드 및 연산에 대한 Built in Function 을 구현한다.

- Node runFunction(FunctionNode func)
  - case 의 추가
    - i CAR
    - ii CDR
    - iii NOT
- Node runBinary(BinaryOpNode node)
  - Binary op node 에 대한 모든 case
- 추가 점수
  - Node runFunction(FunctionNode func)의 나머지 키워드에 대하여 구현하면 추가 점수 부여
    - i CONS
    - ii null?
    - iii atom?
    - iv eq?
    - v cond

2 과제 해결 방법

- Node runFunction(FunctionNode func)
  - 이번에 테스트한 입력문들

```
String source = new String("( cond ( ( > 1 2 ) 0 ) ( #T 1 ) )");
String source = new String("( car '(( 2 3 ) (4 5) 6))");
String source = new String("( cdr '((2 3) (4 5) 6))");
String source = new String("( not #F )");
String source = new String("( cons 1 '( 2 3 4 ) )");
String source = new String("( null? ())");
String source = new String("( atom? 'a ) ");

 String source = new String("( eq? 'a 'a )");
```

  - CAR

```
case CAR:
    if(checkQuote(rhs1)) {
        ListNode listNode = (ListNode)runQuote((ListNode)rhs1);
        return copyNode(listNode.value, CopyMode.NO_NEXT);
    } else {
        return null;
    }
```

    - ◆ List 의 맨 처음 원소를 리턴한다.
    - ◆ List 의 나머지 원소는 출력하지 않는다.
      - i 키워드를 전달받게되는 func 의 다음노드인 rhs1 이 quote node 인지 확인한다.
      - ii quote node 라면, quote 를 제거하고그 다음에 오는 List node 의 값의 value 값을 전달한다.
      - iii enum 의 CopyMode 의 NO_NEXT 값을 copyNode() 메소드에 전달하게 되면, value 값만 다시 타입을 체크하고 리턴하게 된다.
  - CDR

```
case CDR:
    if(checkQuote(rhs1)) {
        ListNode listNode = (ListNode)runQuote((ListNode)rhs1);
        ListNode cdrListNode = new ListNode();
        cdrListNode.value = copyNode(listNode.value.getNext(), CopyMode.NEXT);

        return cdrListNode;
    } else {
        return null;
    }
```

    - ◆ List 의 맨 처음 원소를 제외한 나머지 원소들을 리턴한다.
      - i rhs1 이 quote node 인지를 확인하고 quote node 라면 제거한다.

ii quote node 를 제거하고 남은 나머지 노드들(리스트 노드)의 value 의 next 값(List 의 맨 처음 원소의 다음 원소값)과 CopyMode 의 NEXT 값을 전달하여 노드의 다음 값이 null 값 나올때까지 노드의 타입을 검색하여 리턴한다.

iii 리턴한 노드의 값들을 새로운 ListNode 의 value 로 지정한다.

iv 새롭게 만들어진 ListNode 를 넘긴다.

- NOT

```
case NOT:
    if(rhs1.equals(FALSE_NODE)) {
        return TRUE_NODE;
    } else {
        return FALSE_NODE;
    }
```

i rhs1 의 값이 FALSE_NODE(부정)이라면 반대값인 TRUE_NODE(참)을 리턴하고, rhs1 의 값이 참값이라면 반대 값인 부정값을 리턴한다.

- CONS

```
case CONS:
    ListNode newListNode = new ListNode();

    if(checkQuote(rhs1)) {
        ListNode headNode = (ListNode)runQuote((ListNode)rhs1); //쿼트만 벗겨내면

        if(checkQuote(rhs2)) {
            ListNode nextListNode = (ListNode)runQuote((ListNode)rhs2);
             headNode.value.setNext(nextListNode.value);
            newListNode.value = headNode.value;
        } else {
            headNode.setNext(rhs2);
            newListNode.value = headNode;
        }
    } else if (checkQuote(rhs2)) {
        ListNode nextListNode = (ListNode)runQuote((ListNode)rhs2);
        rhs1.setNext(nextListNode.value);
        newListNode.value = rhs1;
    }
    return newListNode;
```

◆ 한 개의 원소(head)와 한 개의 리스트(tail)를 붙여서 새로운 리스트를 만들어 반환한다.

i rhs1 값이 quote node 라면 quote node 를 runQuote() 메소드를 이용하여 제거하고 새로운 ListNode 인 HeadNode 를 만들어 해당 값을

저장한다.

ii rhs1 의 다음 값인 rhs2 값도 quote node 라면, runQuote()메소드를 이용하여 제거하고, 이 값들을 HeadNode 의 value 값의 다음 값으로 저장하여 두 개의 리스트들을 하나로 만들어준다.

iii rhs2 가 quote node 가 아니라면, headNode 의 다음 값으로 rhs2 를 저장하고 리턴한다.

iv rhs1 이 quote node 가 아니고, rhs2 가 quote node 라면, rhs2 의 quote node 를 제거하고, rhs1 의 다음 값으로 rhs2 의 value 값을 지정해주고 리턴한다.

- null?

```
case NULL_Q:
    if( rhs1 instanceof ListNode) {
        if(rhs2 == null) {
            return TRUE_NODE;
        } else {
            return FALSE_NODE;
        }

    }
```

◆ 리스트가 비어있는 리스트인지 검사한다.

i null? 키워드 다음에 오는 rhs1 이 ListNode 일 때,

ii rhs1 다음에 오는 rhs2 가 바로 null 값을 나타내는 닫는괄호 “)“ 라면, 비어있으므로 TRUE_NODE 반환.

iii 그렇지 않다면 FALSE_NODE 반환

- atom?

```
case ATOM_Q:
    if(checkQuote(rhs1)) {
        if (rhs2 instanceof ListNode) {
            if(rhs2.getNext() == null) {
                return FALSE_NODE;
            }
        } else {
            return TRUE_NODE;
        }
    } else {
        if (rhs1 instanceof ListNode) {
            if(rhs2 == null) {
                return FALSE_NODE;
            }
        } else {
            return TRUE_NODE;
        }
    }
```

◆ 리스트를 검사하는 함수.

◆ list = false, list 가 아니라면 true 반환.

i 키워드 다음 노드인 rhs1 이 quote node 인지 확인한다.

ii rhs1 의 다음 노드인 rhs2 가 ListNode 이고, 바로 다음 값이 닫는 괄호 “) “ 라면 FALSE_NODE 반환, 그렇지 않으면 TRUE_NODE 반환한다.

iii rhs1 이 quote node 가 아니고 listNode 일 때,

iv rhs2 가 닫는 괄호 “) “ 라면 FALSE_NODE 반환. 아니라면 TRUE_NODE 를 반환한다.

- eq?

```
case EQ_Q:
    Node a, b;
    if(checkQuote(rhs1)) {
        a = runQuote((ListNode)rhs1);
    } else {
        a = copyNode(rhs1, CopyMode.NO_NEXT);
    }
    if(checkQuote(rhs2)) {
        b = runQuote((ListNode)rhs2);
    } else {
        b = copyNode(rhs2, CopyMode.NO_NEXT);
    }

    if (a.equals(b)) {
        return TRUE_NODE;
    } else {
        return FALSE_NODE;
    }
```

◆ 두 노드의 값이 같은지 비교한 값을 반환

i rhs1 과 rhs2 에 각각 quote node 가 달려있는지 확인을 하고, 달려있다면 제거해준다.

ii 달려있지 않다면 비교하고 싶은 하나의 값을 위해 copyNode()메소드를 실행하고, 하나의 값만 따로 저장하기 위해 CopyMode 의 NO_NEXT 값을 저장한다.

iii equals()메소드를 이용하여 비교를 한다.

iv 같으면 TRUE_NODE, 아니면 FALSE_NODE 를 반환한다.

- cond

```
        case COND:
                return runCond(rhs1);
        default:
            return null;
        }
    }
    private Node runCond(Node param){
        Node condValue;
        Node runNode = runExpr(((ListNode)param).value);
        if(((BooleanNode)runNode).value == TRUE_NODE.value){
            condValue = ((ListNode)param).value.getNext();
        } else {
            condValue = runCond(param.getNext());
        }
        return condValue;
    }
```

- 조건문
- 키워드 다음에 오는 rhs1 과 rhs2 가 각각의 조건이며, 조건문에서 조건은 계속해서 추가할 수 있다.
- 조건문에서 조건이 충족된다면, 조건의다음값으로 실행 값이 오게된다.
  - i rhs1 의 value 값을 조건이 참인지 거짓인지 판단해야하기 때문에, value 값을 parameter 로 넘겨주는 runExpr()메소드를 통하여 참 거짓 여부를 리턴받는다.
  - ii 리턴받은 값이 참이라면, 조건식의 다음값인 실행값을 리턴한다.
  - iii 그렇지 않다면 runCond()메소드를 재귀적으로 실행한다.

- Node runBinary(BinaryOpNode node)

```
switch(node.value){
case MINUS:
    IntNode minusNode1 = (IntNode) first_param;
    IntNode minusNode2 = (IntNode) second_param;
    IntNode resultMinusNode = new IntNode();
    resultMinusNode.value = minusNode1.value - minusNode2.value;
    result = resultMinusNode;
    return result;
case PLUS:
    IntNode plusNode1 = (IntNode) first_param;
    IntNode plusNode2 = (IntNode) second_param;
    IntNode resultPlusNode = new IntNode();
    resultPlusNode.value = plusNode1.value + plusNode2.value;
    result = resultPlusNode;
    return result;
case DIV:
    IntNode divNode1 = (IntNode) first_param;
    IntNode divNode2 = (IntNode) second_param;
    IntNode resultDivNode = new IntNode();
    resultDivNode.value = divNode1.value / divNode2.value;
    result = resultDivNode;
    return result;
case TIMES:
    IntNode timesNode1 = (IntNode) first_param;
    IntNode timesNode2 = (IntNode) second_param;
    IntNode resultTimesNode = new IntNode();
    resultTimesNode.value = timesNode1.value * timesNode2.value;
    result = resultTimesNode;
    return result;
case GT: //>
    IntNode gtNode1 = (IntNode) first_param;
    IntNode gtNode2 = (IntNode) second_param;

        if (gtNode1.value > gtNode2.value){
            result = TRUE_NODE;
        } else {
            result = FALSE_NODE;
        }
        return result;
    case LT: //<
        IntNode ltNode1 = (IntNode) first_param;
        IntNode ltNode2 = (IntNode) second_param;

        if (ltNode1.value < ltNode2.value){
            result = TRUE_NODE;
        } else {
            result = FALSE_NODE;
        }
        return result;
    default:
        return null;
    }
}
```

◆ 연산자들에 대한 case 구현

i -, +, /, *, >, < 연산자들을 구현한다.

ii first_param 과 second_param 을 각각 연산자에 만든 연산을 실행해준 후 해당 값을 저장한 노드를 리턴해준다.

201000287 일어일문학과 유다훈

3 과제를 해결하면서 느낀 점

1. 구현해야할 문법들이 많고, 어떻게 구현하면 좋을지 감이 잡히질 않아서 시간이 많이 걸렸다.
2. cond 키워드는 조건이 몇 개든지 올 수 있다는 조교의 설명에 의해, 재귀적으로 구현한다는 것을 알게 되었고, 재귀적인 구현 방법 때문에 코드를 상당히 짧게 구현할 수 있었다.
3. atom? 키워드의 경우, 수업 자료의 설명이 잘못 되어있는 부분이 있어서 많이 헷갈렸다.
   - 예시의 (atom? 'a)의 경우는 list 가 아니므로 true 를 반환해야하나 수업자료에서는 false 로 표기되어 있음.