**2023학년도 1학기 [프로그래밍언어론]**

**Lab01 과제 보고서**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 제출일 | 3월 22일 | 분반 | 002 | 학과 | 소프트웨어학부  컴퓨터과학전공 |
| 학번 | 2016133 | 이름 | 이유진 | | |

1. 구현한 프로그램에 대한 설명

- line43: command()의 result변수 초기화에 할당하는 값을 수정함

- line45: 결과 출력문이 예시 사진과 같도록 출력 문구를 수정함

- line49: tip에 제시된 expr()를 EBNF 문법의 { } 범위에 주의하여 수정하고 추가하여 기능을 구현함.

- line86: bexp()에 비교연산자의 연산 기능을 구현함

- line117: relop()에 입력 받는 비교연산자를 return할 수 있도록 구현함

- line151: 기존에 제시된 aexp()에 ‘-‘ 연산 기능을 추가함

- line167: 기존에 제시된 term()에 ‘/’ 연산 기능을 추가함

2. 실행 결과 캡처 사진

텍스트이(가) 표시된 사진

자동 생성된 설명

3. 프로그램 소스 코드 (복사-붙여넣기)

*//소프트웨어학부 컴퓨터과학전공 2016133 이유진*

*//2023-1학기 프로그래밍언어론(002) Lab01*

**import** **java.io.\***;

**class** **Calc** {

int token; int value; int ch;

**private** PushbackInputStream input;

**final** int NUMBER=256;

Calc(PushbackInputStream is){

input = is;

}

int getToken( ) { */\* tokens are characters \*/*

**while** (**true**) {

**try**{

ch = input.read();

**if** (ch == ' ' || ch == '\t' || ch == '\r');

**else** **if** (Character.isDigit(ch)) {

value = number( );

input.unread(ch);

**return** NUMBER;

}

**else** **return** ch;

} **catch** (IOException e) {

System.err.println(e);

}

}

}

void error(){

System.out.printf("parse error: %d\n",ch);

*//System.exit(1);*

}

void match(int c) {

**if** (token == c) token = getToken();

**else** error();

}

void command() {

*/\* command -> expr '\n' \*/*

Object result = expr(); *//수정함*

**if** (token == '\n') */\* end the parse and print the result \*/*

System.out.println("The result is: " + result); *//수정함*

**else** error();

}

Object expr() {*//*

*/\* expr -> bexp {'&' bexp | '|' bexp} | !expr | true | false \*/*

Object result = "";

**if** (token == '!'){*// !<expr>*

match('!');

result = !(boolean)expr();

}

**else** **if** (token == 't') {*// true*

match('t');

result = **true** ;

}

**else** **if** (token == 'f') {*// false*

match('f');

result = **false** ;

}

**else** {*// &<bexp>, |<bexp>*

result = bexp();

**while** (token == '&') {

match('&');

Object temp = bexp();

**if** ((boolean)result == **true** && ((boolean)temp) == **true**) {

result = **true**;

}

**else** result = **false**;

}

**while** (token == '|'){

match('|');

Object temp = bexp();

**if** ((boolean)result == **true** || ((boolean)temp) == **true**) {

result = **true**;

}

**else** result = **false**;

}

}

**return** result;

}

Object bexp(){

*/\* bexp -> aexp [relop aexp] \*/*

Object result = aexp();

String op = relop();

**if** (op == "==") {*// ==*

**if** ((Integer)result == aexp()) result = **true**;

**else** result = **false**;

}

**else** **if** (op == "!=") {*// !=*

**if** ((Integer)result != aexp()) result = **true**;

**else** result = **false**;

}

**else** **if** (op == "<") {*// <*

**if** ((Integer)result < aexp()) result = **true**;

**else** result = **false**;

}

**else** **if** (op == ">") {*// >*

**if** ((Integer)result > aexp()) result = **true**;

**else** result = **false**;

}

**else** **if** (op == "<=") {*// <=*

**if** ((Integer)result <= aexp()) result = **true**;

**else** result = **false**;

}

**else** **if** (op == ">=") {*// >=*

**if** ((Integer)result >= aexp()) result = **true**;

**else** result = **false**;

}

**return** result;

}

String relop(){*//done*

*/\* == | != | < | > | <= | >= \*/*

String result = "";

**while** (token == '='){*// ==*

match('=');

result = "==";

}

**if** (token == '!') {

match('!');

**if** (token == '=') {*// !=*

match('=');

result = "!=";

}

**else** result = "!";*// !*

}

**else** **if** (token == '<'){

match('<');

**if** (token == '='){*// <=*

match('=');

result = "<=";

}

**else** result = "<";*// <*

}

**else** **if** (token == '>'){

match('>');

**if** (token == '='){*// >=*

match('=');

result = ">=";

}

**else** result = ">";*// >*

}

**return** result;

}

int aexp(){*//done*

*/\* aexp -> term { '+' term | '-' term } \*/*

int result = term();

**while** (token == '+' || token == '-') {

**if** (token == '+') {*// +<term>*

match('+');

result += term();

}

**else** {*// -<term>*

match('-');

result -= term();

}

}

**return** result;

}

int term( ) {*//done*

*/\* term -> factor { '\*' factor | '/' factor} \*/*

int result = factor();

**while** (token == '\*' || token == '/' ) {

**if** (token == '\*') {*// \*<factor>*

match('\*');

result \*= factor();

}

**else** {*// /<factor>*

match('/');

result /= factor();

}

}

**return** result;

}

int factor(){

*/\* factor -> [-] ( number | '('aexp')' ) \*/*

int result = 0;

**if** (token == '(') {

match('(');

result = aexp();

match(')');

}

**else** **if** (token == NUMBER){

result = value;

match(NUMBER); *//token = getToken();*

}

**return** result;

}

**private** int number( ) {

*/\* number -> digit { digit } \*/*

int result = ch - '0';

**try**{

ch = input.read();

**while** (Character.isDigit(ch)) {

result = 10 \* result + ch - '0';

ch = input.read();

}

} **catch** (IOException e) {

System.err.println(e);

}

**return** result;

}

void parse(){

token = getToken(); *//get the first token*

command(); *//call the parsing command*

}

**public** **static** void main(String []args){

Calc calc = **new** Calc(**new** PushbackInputStream(System.in));

**while** (**true**){

System.out.print(">> ");

calc.parse();

}

}

}