```
node * parse_expression(lexan & LEX)
{ LEX.nextlex();
  if (LEX.kind == LX_number)
    return new node(N_integer, LEX.intvalue);

  else if (LEX.kind == LX_variable)
    return new node(N_variable, LEX.syminfo);

  else if (LEX.kind == LX_openround)
  { node * r = new node(N_binaryexp);
    r->subtree.push_back(parse_expression(LEX));
    LEX.nextlex();
    if (! isoperator(LEX.kind))
      LEX.error("expected operator, found " + LEX.form);
    r->intvalue = (int)LEX.kind;
    r->subtree.push_back(parse_expression(LEX));
    LEX.nextlex();
    if (LEX.kind != LX_closeround)
      LEX.error("expected closing parenthesis, found " + LEX.form);
    return r; }

  else
    LEX.error("expecting expression, found " + LEX.form); }


node * parse_statement(lexan & LEX)
{ LEX.nextlex();
  if (LEX.kind == LX_RW_print)
  { node * r = new node(N_printstmt);
    r->subtree.push_back(parse_expression(LEX));
    return r; }

  else if (LEX.kind == LX_RW_when)
  { node * r = new node(N_whenstmt);
    r->subtree.push_back(parse_expression(LEX));
    r->subtree.push_back(parse_statement(LEX));
    return r; }

  else if (LEX.kind == LX_RW_var)
  { node * r = new node(N_vardecl);
    LEX.nextlex();
    if (LEX.kind != LX_variable)
      LEX.error("in assignment, expecting variable, found " + LEX.form);
    r->syminfo = LEX.syminfo;
    LEX.nextlex();
    if (LEX.kind != LX_OP_assign)
      LEX.error("in assignment, expecting =, found " + LEX.form);
    r->subtree.push_back(parse_expression(LEX));
    return r; }

  else
    LEX.error("expecting statement, found " + LEX.form); }
```

```
$ lang3
var x = (((1+2)*(3+4))-(cat * 1234))
vardecl syminfo=0x804e070
   binaryexp intvalue=4
      binaryexp intvalue=5
         binaryexp intvalue=3
            integer intvalue=1
            integer intvalue=2
         binaryexp intvalue=3
            integer intvalue=3
            integer intvalue=4
      binaryexp intvalue=5
         variable syminfo=0x804e0d0
         integer intvalue=1234
```