编译器构造实验

Lab5—实验 1

熟悉 Oberon-0 语言定义

姓名：郝裕玮

班级：计科 1 班

学号：18329015

目录

1 编写一个正确的 Oberon-0 源程序

```
MODULE Test;
    (* 计算阶乘 *)
    PROCEDURE Factorial;
        VAR n ,result: INTEGER;
    BEGIN
        result := 1;
        Read(n);
        IF n = 0 THEN
            result := 1;
        END
        WHILE n >= 1 DO
            result := n * result;
            n := n - 1
        END;
        Write(result); WriteLn
    END Factorial;

    (* 计算两数之和与两数之差 *)
    PROCEDURE AddSub;
    TYPE
        res = INTEGER;
        sum = RECORD
            a, b : INTEGER;
        minus = RECORD
            c, d : INTEGER;
        END;
    VAR
        add: ARRAY 1 OF sum;
        sub: ARRAY 1 OF minus;
        addres, subres: res;
    BEGIN
        READ(add[0].a);
        READ(add[0].b);
        READ(sub[0].c);
        READ(sub[0].d);
        addres := add[0].a + add[0].b;
        subres := sub[0].c - sub[0].d;
        Write(addres); Write(subres); WriteLn
    END AddSub;
END Test.
```

2 编写上述 Oberon-0 源程序的变异程序

词法错误：

（1）IllegalSymbolException

当识别一个单词时遇到不合法的输入符号（譬如@、 $等符号）则抛出该异常。

```
(* IllegalSymbolException *)
MODULE Test;
    (* 计算阶乘 *)
    PROCEDURE Factorial;
        VAR n ,result: INTEGER;
    BEGIN
        re@sult := 1;
        Read(n);
```

（2）IllegalIntegerException

当整数常量（无论是十进制还是八进制） 与其后的标识符之间无空白分隔时抛出该异常。

```
(* IllegalIntegerException *)
MODULE Test;
    (* 计算阶乘 *)
    PROCEDURE Factorial;
        VAR 1n ,result: INTEGER;
    BEGIN
```

（3）IllegalIntegerRangeException

当识别出的整数常量（无论是十进制还是八进制）大于本文档约定的整数常量值最大限制时抛出此异常。

```
(* IllegalIntegerRangeException *)
MODULE Test;
    (* 计算阶乘 *)
    PROCEDURE Factorial;
        VAR n ,result: INTEGER;
    BEGIN
        result := 1111111111111;
        Read(n);
```

（4）IllegalOctalException

当 0 开头的整数常量中含有 0～7 之外的符号（包括 8 和 9）时抛出该异常。

```
(* IllegalOctalException *)
MODULE Test;
    (* 计算阶乘 *)
    PROCEDURE Factorial;
        VAR n ,result: INTEGER;
    BEGIN
        result := 09;
        Read(n);
        IF n = 0 THEN
```

（5）IllegalIdentifierLengthException

当识别出的一个标识符长度超过最大限制时抛出该异常。

```
(* IllegalIdentifierLengthException *)
MODULE Test;
    (* 计算阶乘 *)
    PROCEDURE Factorial;
        VAR n ,result: INTEGER;
    BEGIN
        resultabcdefghijklmnopqrstuvwxyz := 1;
        Read(n);
        IF n = 0 THEN
            result := 1;
        END
```

（6）MismatchedCommentException

当“(*”开头的注释直至扫描到最后一个符号都找不到配对的“*)”时抛出该异常。

```
(* 计算两数之和与两数之差
PROCEDURE AddSub;
TYPE
    res = INTEGER;
    sum = RECORD
        a, b : INTEGER;
    minus = RECORD
        c, d : INTEGER;
    END;
VAR
```

语法错误：

（1）MissingLeftParenthesisException

当分析到左右圆括号不匹配、且缺少左括号时则抛出该异常。

```
(* MissingLeftParenthesisException *)
MODULE Test;
    (* 计算阶乘 *)
    PROCEDURE Factorial;
        VAR n ,result: INTEGER;
    BEGIN
        result := 1;
        Readn);
```

（2）MissingRightParenthesisException

当分析到左右圆括号不匹配、且缺少右括号时则抛出该异常。

```
(* MissingRightParenthesisException *)
MODULE Test;
    (* 计算阶乘 *)
    PROCEDURE Factorial;
        VAR n ,result: INTEGER;
    BEGIN
        result := 1;
        Read(n;
```

（3）MissingOperatorException

当分析到程序中缺少运算符时，或调用预定义函数缺少相应的参数时则抛出该异常。

```
BEGIN
    READ(add[0].a);
    READ(add[0].b);
    READ(sub[0].c);
    READ(sub[0].d);
    addres := add[0].a  add[0].b;
    subres := sub[0].c - sub[0].d;
    Write(addres); Write(subres); WriteLn
END AddSub;
```

（4）MissingOperandException

当分析到程序中缺少操作数时则抛出该异常。

```
(* MissingOperandException *)
MODULE Test;
    (* 计算阶乘 *)
    PROCEDURE Factorial;
        VAR n ,result: INTEGER;
    BEGIN
        result := 1;
        Read(n);
        IF n = 0 THEN
            result := 1;
        END
        WHILE n >= 1 DO
            result := n * result;
            n := n -
        END;
```

语义错误：

（1）TypeMismatchedException

当分析到表达式、赋值语句、或参数传递等构造中出现类型不兼容错误时则抛出该异常

```
(* TypeMismatchedException *)
MODULE Test;
    (* 计算阶乘 *)
    PROCEDURE Factorial;
        VAR n ,result: BOOLEAN;
    BEGIN
```

（2）ParameterMismatchedException

当分析到过程调用的实际参数数目与过程声明的形式参数数目不一致时则抛出该异常。 注意，当实际参数的数目与形式参数一致，

只是存在某一参数的类型不兼容时，应抛出TypeMismatchedException 异常而不是抛出本异常。

```
(* 计算阶乘 *)
PROCEDURE Factorial;
    VAR n ,result: INTEGER;
BEGIN
    result := 1;
    Read(n);
    IF n = 0 THEN
        result := 1;
    END
    WHILE n >= 1 DO
        result := n * result;
        n := n - 1
    END;
    Write(result); WriteLn
END Factorial;
```

```
BEGIN
    Factorial(1,2,3);
END AddSub;
```

3 讨论 Oberon-0 语言的特点

保留字与关键字的区别：

（1）保留字是程序中预先定义的特殊的字或词，程序员不能再将这些字作为变量名或者过程名使用。主要用于对不同程序块进行组织划分。

（2）关键字是编译器保留使用的字或词，主要用于提供部分预定的函数和变量。

Oberon-0 与 JAVA、C/C++的区别：

（1）Oberon-0 不区分变量名大小写。JAVA、C/C++区分变量名大小写。

（2）Oberon-0 的不同过程 PROCEDURE 使用 BEGIN 和 END 来区分。Java、C/C++中不同代码块用{}来区分。

（3）Oberon-0 语言不支持实数除法运算（仅支持整除运算）。JAVA、C/C++支持实数除法运算（因为存在浮点数）。

（4）Oberon-0 没有主函数 main。但 JAVA、C/C++必须有主函数 main。

（5）Oberon-0 的变量声明是变量类型写在变量名之后。Java、C/C++的变量声明是变量类型写在变量名之前。

4 讨论 Oberon-0 文法定义的二义性

与 Java、 C/C++相比，Oberon-0 文法不存在二义性。

（1）Java、 C/C++的 if-else 二义性问题

Oberon-0 使用 END 标识来解决该问题。每个 IF 语句都需要用 END 来标志该语句结束，这样就不存在多个 if-else 的匹配问题。

（2）Java、 C/C++的算术表达式中运算符号的优先级问题

Oberon-0 在文法中规定了各种运算符的优先级，从而避免了一个表达式产生多个语法树的二义性问题。

5 实验心得

该实验让我对 Oberon-0 语言有了初步的认识，并可以根据已有的语法规则去编写正确的相关程序代码。同时通过对比，我对二义性也有了更深刻的认识。