实验一、熟悉 Oberon-0 语言定义

19335019 陈泷

实验步骤 1.1、 编写一个正确的 Oberon-0 源程序

遵循 Oberon-0 语言的 BNF 定义，编写了一个 Oberon-0 源程序。这个源程序的内容是定义账户类型，寻找n个账户中薪资的最大值。在这个源程序中，用到了 Oberon-0语言的所有语法构造，即程序覆盖了 Oberon-0 语言提供的模块、声明（类型、常量、变量等）、 过程声明与调用、 语句、表达式等各种构造。

源程序代码如下：

```
MODULE program;
    (* 定义账户类型 *)
    TYPE
        AccountRecord = RECORD
            id: INTEGER;
            salary: INTEGER
        END;

    (* 寻找n个账户中薪资的最大值 *)
    PROCEDURE FindMax;
        VAR x, n, i: INTEGER;
            accounts: ARRAY 100 of AccountRecord;
    BEGIN
        Read(n);
        i := 1;
        x := 0;
        if n > 0 THEN
            accounts[0].id = 0;
            x := Read(accounts[0].salary)
        END;
        WHILE ( i < n & i < 100 ) DO
            accounts[i].id = i;
            Read(accounts[i].salary);
            if accounts[i].salary > x THEN
                x := accounts[i].salary
            ELSE
                x := x
            END;
            i := i + 1;
        END;
        Write(x); Write(n); Write(i); WriteLn
    END FindMax;
END program.
```

实验步骤 1.2、 编写上述 Oberon-0 源程序的变异程序

结合之前实验中Exception的错误类型生成以下变异程序：

1. Illegal Symbol Exception：识别单词时遇到不合法的输入符号(譬如@、 $等符号)则抛出该异常。
   line 6 改为：`sal$ary: INTEGER`
2. Parameter Mismatched Exception：当调用的函数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。
   line 32和33之间添加：

```
PROCEDURE call();
    BEGIN
        Findmax(5);
    END call;
```

3. Illegal Octal Exception：当 0 开头的整数常量中含有 0~7 之外的符号(包括 8 和 9或其他字母)时抛出该异常。
   line 15 改为：`i := 08;`
4. Missing Right Parenthesis Exception：当左右圆括号不匹配、且缺少右括号时则抛出该异常。
   line 24 改为：`if (accounts[i].salary >= x THEN`
5. Missing Right Parenthesis Exception：当左右圆括号不匹配、且缺少左括号时则抛出该异常。
   line 24 改为：`if accounts[i].salary >= x) THEN`
6. Type Mismatched Exception：当表达式、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。
   line 27 改为：`x := ~x`
7. 含有不合法的常量：常量中数字的个数超出限制
   line 15 改为：`i := 10000000000000000000000000000001;`
8. 出现实数除法运算
   line 15 改为：`i := 1 div 3;`
9. Divided By Zero Exception: 除数为 0 错误
   line 15 改为：`i := 1 div 0;`
10. 标识符长度不允许超过 24 个字符
    line11改为：`VAR x, n, izxcvbnmasdfghjklqwert: INTEGER;`

实验步骤 1.3、 讨论 Oberon-0 语言的特点

保留字和关键字的区别

保留字主要用于划分程序的组织结构。保留字是程序预先定义的一些字符或字符串，其含义已经预先定义，用户不能将其作为其他变量或是标识符来使用。这样的设计有助于组织程序语言的层次架构。

关键字主要用于实现程序的功能。关键字被编译器保留使用，一般包括数据类型和程序流控制，预定义的变量和函数等，更侧重于实现程序的功能。

与 Java、 C/C++等常见语言的表达式不同之处

1. 声明变量时，Oberon-0先给出变量名再给出类型；C/C++或Java是先给出变量类型再给出变量名。
2. Oberon-0是大小写无关的；C/C++或Java是大小写相关的。
3. Oberon-0有 type、begin、end等保留字划分程序结构；C/C++或Java是没有的。
4. Oberon-0不支持 FOR 和 DO-WHILE 等保留字；C/C++或Java是支持的。
5. Oberon-0只有两种数据类型，且不支持类型转换；C/C++或Java拥有更多的数据类型，且支持类型转换。

6. Oberon-0通过限制常量中数字的个数来限制大小；C/C++或Java是通过规定特定类型占有的字节数来限制大小。
7. Oberon-0可以没有main函数；C/C++或Java必须有main函数。

实验步骤 1.4、 讨论 Oberon-0 文法定义的二义性

我认为Oberon-0文法没有二义性。

在其他高级程序设计语言中常见的那些二义性问题在Oberon-0 语言中并未出现的原因：

- 处理运算法运算的二义性：通过多个产生式规定了不同运算符的优先级和结合性，从二避免了二义性。
- 处理if-else的二义性：通过BEGIN、END 表示符显式确定了各个关键字的作用范围，从二避免了二义性。

实验心得与体会

通过实验一，我熟悉 Oberon-0 语言定义，并且通过与C/C++或Java的比较加深了对几种语言的认识与辩证性思考，体会到不同的语言设计对实际编程的影响。另外通过阅读课本以及有关资料，也加深了对语法二义性、保留字和关键字区别等内容的理解。总之，在这一过程中有一定的收获。