EX 1

20308003 曾伟超

编写程序

一个最好展示用的例子是使用递归的，这样必然会使用到函数调用，自然会联想到递归调用，而递归一个最初级的例子就是斐波那契数列，而剩下没用到的部分，则只能使用下面的 `test` 来做补充了，具体见 `testcases` 目录

之后是变异程序

001：词法错误，不合法的标识符 (n@)

002：词法错误，不支持的八进制数 (019)

003：缺失左括号

004：缺失右括号

005：INTEGER 使用了逻辑运算

006：过程调用的时候左右值错误

007：MODULE END 和其名字不搭

008：`CONST` 定于缺失操作符

009：`TYPE` 定义缺失操作符

010：过程调用缺失右括号

011：过程调用缺失左括号

012：未闭合的注释

013：不合法的整数 (0b)

014：整数范围异常

015：不合法的标识符长度

016：`VAR` 定义部分缺失类型

017：过程形参缺失类型

018：`RECORD` 定义缺失类型

019：数组长度未定义

020：布尔表达式缺失操作数

021：传参过程缺失操作符

022：布尔表达式缺失操作数

023：布尔表达式缺失操作数

024：表达式缺失左括号

025：表达式类型不匹配

026：过程调用的时候左右值错误

027：未定义的类型

028：整数使用选择符错误(INTEGER.val)

029：将布尔表达式赋值给整数

030：尝试调用一个不存在的过程

关键字 VS 保留字

实验文档所定义的关键字和保留字似乎与龙书对这些的定义有些许出入，这里以实验文档的来说，自行总结了一下

保留字，如 `IF`，`THEN`，`ELSIF` 在后面会参与到语法的构成中，作为语法中的一个环节出现，其具体的语义是明确定义的，主要会对程序的组织有影响，例如控制流的转移等

关键字，如 `INTEGER`，`WRITE`，`WRITELN`，其不会参与到语法的构成中，主要会对一些功能性的部分产生影响

语言特点

通过阅读 Oberon-0 的 EBNF 语法定义，可以知道，Oberon 和 C 语言类似，是一个面向过程的编程语言，但在一些地方与 C 语言有着明显的区别，具体的一些特点如下

1. 面向过程设计，没有类的概念
2. 支持子程序，即可以进行过程调用，且传参有按引用传递和按值传递两种方式，按需选择
3. 有模块的概念，且一个模块可以没有主程序，即只是一串的函数，变量声明，是可行的
4. 支持将标识符声明为局部于某一过程，即局部变量
5. 大小写不敏感，这点与 C 有显著区别，C，Java 等语言是大小写敏感的
6. 支持注释，使用 `(*` 进行包裹 `*)`
7. 类型只有无符号整数和布尔，支持八进制(以 0 开头)和十进制(非 0 开头)，不支持 TRUE，FALSE 的布尔常量，不支持布尔和整数之间的转换
8. 允许进行类型声明，即使用某一标识符来作为类型，类似 C 中的 `typedef`
9. 有类似结构体的 `RECORD`
10. 允许类型的嵌套，例如数组的数组，或者是结构体的数组
11. 有顺序，选择，循环的结构，但循环只有 WHILE 循环，C/C++，Java 还会有 `for` 循环和 `do-while` 循环的结构
12. 变量定义的时候，类型通过 `:` 跟在变量名后面
13. 变量必须在过程的最开头定义，而不是可以在任意位置定义
14. 没有 RETURN 语句，过程返回值只能通过参数引用返回
15. 和 C/C++，Java 一样，拥有常量

和 C/C++，Java 等的一些主要区别为

1. C/C++，Java 大小写敏感
2. C/C++，Java 类型更为丰富，例如 `char` 字符类型，有符号数字类型，浮点数类型等
3. 在 C 中，仅支持按值传递一种方式，要想有副作用，需要使用指针
4. 循环的结构较为单一，只有 WHILE 循环
5. 不支持类型的转换

6. 类型在前，而不是通过 `:` 的方式跟在后面
7. 可以在过程的任意位置定义变量

文法二义性

通过阅读其 EBNF 语法定义，可以知道其是无二义性的

高级编程语言中一个最常见的二义性在于悬空 `else`，拿 C 举例，下面的代码是存在二义性的

```
if (a == 1)
    if (b == 2)
        printf("b == 2\n");
else
    printf("a != 1\n");
```

上面代码的 `else` 是存在二义性的，可以认为其所属的 `if` 是 `b == 2` 或者是 `a == 1`，一般来说，`else` 会采取最近匹配，在上述情况中，即认为是属于 `b == 2` 这个 `if`，这就显然是和想要实现的效果所不同的

而 Oberon-0 对于 `if` 选择则是强制要求显式的指定 `END` 来表示 `if` 的终结，实际上，对于上面的代码，如果有 `{}` 的指定，如下

```
if (a == 1) {
    if (b == 2) {
        printf("b == 2\n");
    }
}
else
    printf("a != 1\n");
```

会发现也能够消除文法方面的二义性

实验心得

通过这一部分的实验，对于 Oberon-0 有了些初步的认识，同时通过实验文档中的一些指引，了解到了编程语言中的保留字和关键字的一些区别，加深了我对于这些概念的认知；同时通过对 Oberon-0 语法的二义性的讨论，了解到现在一些编程语言中二义性产生的原因以及对应的一些可行的解决方法，能够帮助我以后更好的规避这样的写法，采取更为优良的编程习惯