附录 期中考试解析

> 这部分是考试完写的，题目准确率90%，可能有点小问题，仅供参考
>
> 由于其他部分是需要上交的，这部分不用，所以这部分可能就~~夹带点私货~~了

1.题目说明

期中考试增加了两个地方，第一个是在 `Stmt` 增加了文法，第二个是增加了新的 `数字类型`

- 文法增加：
  Stmt → `'repeat'` Stmt `'until'` `'('` Cond `')'`
- 文法修改：
  Number → IntConst | `HexadecimalConst`
- 保留字增加：
  - `repeat` REPEATTK
  - `until` UNTILTK
  - 十六进制数 HEXCON
- 十六进制说明：
  - HexadecimalConst → HexadecimalPrefix HexadecimalDigit | HexadecimalConst HexadecimalDigit
  - HexadecimalPrefix → `'0x'` | `'0X'`
  - HexadecimalDigit → `'0'` | `'1'` | `'2'` | `'3'` | `'4'` | `'5'` | `'6'` | `'7'` | `'8'` | `'9'` | `'A'` | `'B'` | `'C'` | `'D'` | `'E'` | `'F'` | `'a'` | `'b'` | `'c'` | `'d'` | `'e'` | `'f'`

一共十个样例点（这个具体怎么分的忘记了，但是应该不重要）

- **testfile 1-3** 只判断 `源程序`
- **testfile 4-5** 只添加 `repeat/until`
- **testfile 6-8** 只添加 `十六进制`
- **testfile 9-10** 添加 `repeat/until` 和 `十六进制`

2.repeat/until

一般的评分标准都有 `什么都不加的源程序评分` （简称送分题）第一步先直接下下来提交一版，看看能不能过送分题，如果过了，那么就可以开始写代码了；如果没过，那么就要看看源程序哪里出问题（不过应该大概率不会吧QAQ）

首先是词法的修改，即 `repeat` 和 `until` 俩保留字，所以可以先去 `词法分析` 增加两个保留字的识别

（然而我直接偷懒，因为testfile给的一定是**没有错误的文件**，故只要是个词我都扔进Token表中，统一放到Syntax下判断）

接下来是对 `语法分析` 的修改，首先是在 `特殊字符` 表中添加 `repeat` 和 `until` .

```
ReservedCharacter.put("repeat","REPEATTK");
ReservedCharacter.put("until","UNTILTK");
```

然后是在 `Stmt` 中添加文法

```
public void Stmt(){
    if(sym.equals("{")){Block();}
    else if···

    else if(sym.equals("repeat")){nextsym();Stmt();
        if(sym.equals("until")){nextsym();
            if(sym.equals("(")){nextsym();Cond();
                if(sym.equals(")")){nextsym();}
            }
        }
    }

    else if···
    output("<Stmt>");
}
```

然后在所有用到**Stmt的FIRST集**的地方添加**repeat**判断，包含

- `BlockItem` → Decl | `Stmt`
- `Stmt` → 'if' '(' Cond ')' `Stmt` ['else' `Stmt`]
- `Stmt` → 'while' '(' Cond ')' `Stmt`

仔细判断可以发现，后面两个文法中，我们实际写的时候**不需要判断Stmt的首字符**，因为Stmt的前面都是`终结符`，所以不需要作修改。而 `BlockItem` 则需要判断首字符集来判断是 `Decl` 还是 `Stmt` ，所以需要添加 `repeat` 的判断。

然后，我们需要判断哪些地方用到了**BlockItem的首字符**集合

- `Block` → '{' { `BlockItem` } '}'

由于 `BlockItem` 是可以**重复0次或任意次**，故递归下降子程序中，我们要用 `while` 和 `First集合` 判断。所以再这里也需要添加 `repeat` **判断**

所以在上述**两处地方**递归下降子程序中增加

```
||sym.equals("repeat")
```

（**小技巧**：因为只有Stmt里面用到了 `repeat` ，同理，只有Stmt里面用到了 `return` ，所以可以在`IDEA` 中直接 `Ctrl+F` 搜索 `return` ，只要有 `return` 判断的地方，加上 `repeat` 判断就可以了）

此时提交一版，看看过了没有，如果过了，说明这个部分没有问题，可以继续开始下一部分。

3.Hexadecimal

这一部分是十六进制的数字，所以我们同样需要修改 `词法分析` 和 `语法分析` 两部分。

首先，一开始我们只需要判断 `Number` 类型是不是**整型数字**，而此时，Number不仅包含了整形，还包含了**十六进制**的判断，所以我们需要有一个能够判断十六进制的函数。

顺带一提，判断Number的时候，Java有**isDigit**的方法判断字符是否是数字，当然你可以去使用 `正则表达式`（这里附上一篇同为小学期助教的李昊哥哥的博客，感兴趣的可以去学习一下）

这里我们采用最简单的判别方法，首先进行文法的修改，即消除左递归。修改后文法为

- HexadecimalConst → (`0x` | `0X`) HexadecimalDigit { HexadecimalDigit }

首先判断是否是 `0` ，如果是，判断下一个字符是否是 `x/X` ，如果是，那么接下来判断是否是十六进制数，即0-9/a-f/A-F，如果是，则返回true，否则返回false。

```
public static boolean isHexadecimal(String str){
    if(str.length() < 2) { return false; }
    if(str.charAt(0) == '0'){
        if(str.charAt(1) == 'x' || str.charAt(1) == 'X'){
            for(int i = 2; i < str.length(); i++){
                if(!((str.charAt(i) >= '0' && str.charAt(i) <= '9') ||
(str.charAt(i) >= 'a' && str.charAt(i) <= 'f') || (str.charAt(i) >= 'A' &&
str.charAt(i) <= 'F'))){
                    return false;
                }
            }
            return true;
        }
    }
    return false;
}
```

其实还是可以投机取巧，因为给的文法**一定是正确的**，所以甚至只要判断 `0x` 或 `0X` 就可以了，因为**不可能有第二种以0x或0X打头的Token**

所以，假设之前判断是否是整数的函数叫**isNumber()**，那么我们只需要把之前的判断内容换成另一个函数，在原函数下增加一个判断即可，这样递归下降子程序可以不用更改，即

```
public static boolean isInt(String str){
    for (int i=0;i<str.length();i++) {
        if (!Character.isDigit(str.charAt(i))) {
            return false;
        }
    }
    return true;
}
public static boolean isNumber(String str){
    return (isInt(str) || isHexadecimal(str));
}
```

最后，在输出的时候，只需要别忘了判断是输出 `INTCON` 还是 `HEXCON` 即可

```
if(sym.charAt(0)=='<'&&sym.length()>1&&sym.charAt(1)-'A'>=0&&sym.charAt(1)-'A'<26){
    pw.println(sym);
    pw.flush();
}
else if(ReservedCharacter.containsKey(sym)){
    pw.println(ReservedCharacter.get(sym)+" "+sym);
    pw.flush();
}
else{
    if(sym.charAt(0)=='"'){
        pw.println("STRCON "+sym);
        pw.flush();
    }
    else if(isInt(sym)){
        pw.println("INTCON "+sym);
        pw.flush();
    }
    else if(isHexadecimal(sym)){
        pw.println("HEXCON "+sym);
        pw.flush();
    }
    else{
        pw.println("IDENFR "+sym);
        pw.flush();
    }
}
```

至此，十个样例点全部通过，考试完成，如果顺利的话**15-20分钟**就可以完成，所以不用慌张。