用 Lex 设计词法分析器 2

一、实验目的：

学会用 lex 设计一个词法分析器，并考虑其与后续语法分析器的链接问题。

二、实验要求：

1. 要求每次调用词法分析函数 yylex 时，只返回一个记号(token)；
2. 为记号选择适当的属性值，并且每次词法分析函数返回记号前，都将记号的属性值存入全局变量 yylval 中。(yylval 可以自己定义为全局变量)；
3. 记号属性值的选择：标识符的属性为标识符的名字字符串（例如，标识符 name1 的属性为字符串”name1”），整数的属性为整数值，浮点数的属性为浮点数值。其他记号属性值可自己选择。关键字可以省略属性。
4. 注意：由于属性值需要存入 yylval 中，并且记号属性值的类型比较多（可能为字符串、整数、浮点数等），因此 yylval 必须能同时存放各种类型的值（提示：将 yylval 设置为 union 类型）。
5. 在 cygwin 下用 flex 和 gcc 工具将实验调试通过，并能通过例子 parser0 中 testcases 目录下的 test1.p 测试例的测试。

三、具体实现（文件 exam2.1 的展示）

1. 定义段部分新增如下：

```
union abc{
    int intNum;
    float floatNum;
    char *str;
}yylval;

char* installID();
float installFloat();
int installInt();
```

- 定义保存记号属性的 yylval 变量为联合体类型，属性值类型可以为字符串（例如 id），可以为整数和浮点数，yylval 默认是 int 型，不满足要求，因此定义为联合体，根据需要使用，联合体里的成员有三个。
- 下面三个均为辅助函数，声明翻译规则中会使用到，具体功能实现在第三部分辅助函数段完成。

2. 翻译规则部分修改如下：

```
<INITIAL>{id}          {yylval.str = installID(); return (ID);}
<INITIAL>{Int}         {yylval.intNum = installInt(); return (INT);}
<INITIAL>{Float}       {yylval.floatNum = installFloat(); return (FLOAT);}
```

- 若记号是 id，则使得该记号的属性值为 id 本身的字符串。
- 若记号为整形数或浮点数，则记号的属性值为这个数的值。

3. 输出函数修改如下：

```
case INT: fprintf(yyout, "(INT, \"%d\") ", yylval.intNum);break;
case FLOAT: fprintf(yyout, "(FLOAT, \"%f\") ", yylval.floatNum);break;
case ID: fprintf(yyout, "(ID, \"%s\") ", yylval.str);break;
```

- 对于记号为 id、整形数、浮点数的输出，属性值改为用通过 yylval 输出，注意修改输出格式，id 仍然是”%s”，整形数是”%d”，浮点数是”%f”。

4. 辅助函数功能的实现：

```c
char* installID () {
    return yytext;
}

float installFloat () {
    return (float)atof(yytext);
}

int installInt(){
    return (int)atof(yytext);
}
```

- yytext 指向输入缓冲区当前词法单元(lexeme)的第一个字符
- 第一个函数是用于把词法单元为 id 的装入符号表，返回类型为字符数组，因此直接返回 yytext 即可。
- 第二个函数是若词法单元是浮点数，atof 函数的作用是把字符串转换成浮点数，把 yytext 转化为浮点数。
- 第三个函数和第二个同理，需要把浮点数转化为整形。

5. 测试文件 test1.p 内容如下：

```
1   PROGRAM test;
2   VAR i, j, k: INTEGER;
3       f0: REAL;
4   BEGIN
5     i := 1;
6     j := 1;
7     k := 0;
8     f0 := 3.2;
9     WHILE k<=100 DO
10      BEGIN
11        IF j <20 THEN
12          BEGIN
13            j := i;
14            k := k+1;
15            f0 := f0*0.2
16          END
17        ELSE
18          BEGIN
19            j := k;
20            k := k-2;
21            f0 := f0/.2
22          END
23    END
24  END.
```

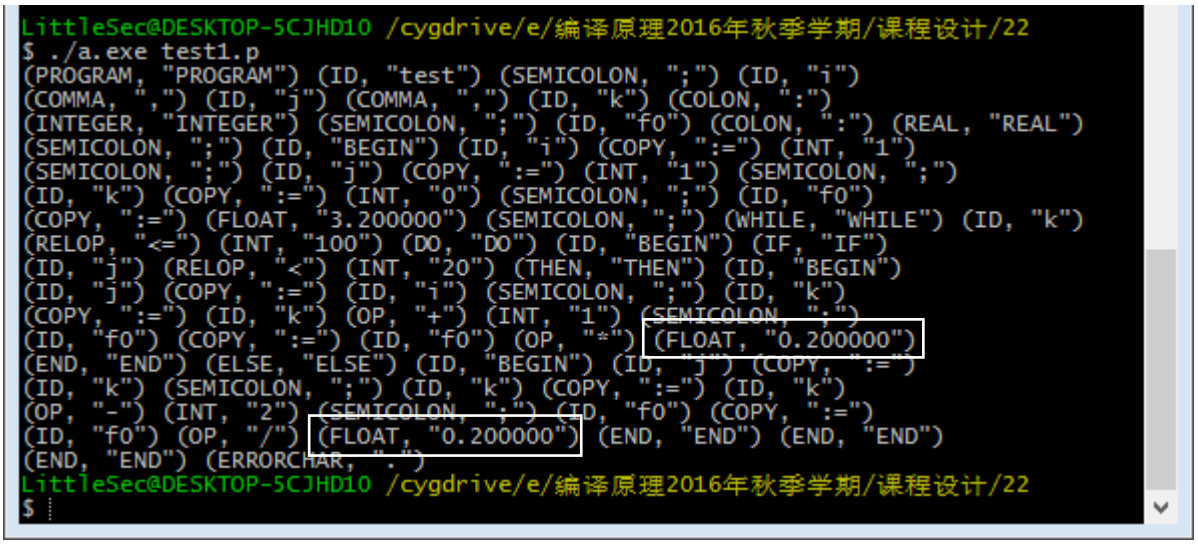
测试文件中包含：

- ✓ PROGRAM、VAR、BEGIN、END、IF、THEN、ELSE、WHILE、INTEGER、REAL 关键字
- ✓ 4 个变量名
- ✓ 赋值语句若干
- ✓ 整型数和浮点数若干
- ✓ 比较表达式若干
- ✓ 运算表达式若干

6. 在 cygwin 下用 flex 编译器对新的 exam2.l 进行编译，生成 lex.yy.c 文件，用 gcc 编译器对该 C 源程序进行编译得到 a.exe 文件，用测试文件 test1.p 进行测试，如图所示：

```
LittleSec@DESKTOP-5CJHD10 /cygdrive/e/编译原理2016年秋季学期/课程设计/11
$ ./a.exe test1.p
(PROGRAM, "PROGRAM") (ID, "test") (SEMICOLON, ";") (ID, "i")
(COMMA, ",") (ID, "j") (COMMA, ",") (ID, "k") (COLON, ":")
(INTEGER, "INTEGER") (SEMICOLON, ";") (ID, "f0") (COLON, ":") (REAL, "REAL")
(SEMICOLON, ";") (ID, "BEGIN") (ID, "i") (COPY, ":=") (INT, "1")
(SEMICOLON, ";") (ID, "j") (COPY, ":=") (INT, "1") (SEMICOLON, ";")
(ID, "k") (COPY, ":=") (INT, "0") (SEMICOLON, ";") (ID, "f0")
(COPY, ":=") (FLOAT, "3.2") (SEMICOLON, ";") (WHILE, "WHILE") (ID, "k")
(RELOP, "<=") (INT, "100") (DO, "DO") (ID, "BEGIN") (IF, "IF")
(ID, "j") (RELOP, "<") (INT, "20") (THEN, "THEN") (ID, "BEGIN")
(ID, "j") (COPY, ":=") (ID, "i") (SEMICOLON, ";") (ID, "k")
(COPY, ":=") (ID, "k") (OP, "+") (INT, "1") (SEMICOLON, ";")
(ID, "f0") (COPY, ":=") (ID, "f0") (OP, "*") (FLOAT, "0.2")
(END, "END") (ELSE, "ELSE") (ID, "BEGIN") (ID, "j") (COPY, ":=")
(ID, "k") (SEMICOLON, ";") (ID, "k") (COPY, ":=") (ID, "k")
(OP, "-") (INT, "2") (SEMICOLON, ";") (ID, "f0") (COPY, ":=")
(ID, "f0") (OP, "/") (FLOAT, ".2") (END, "END") (END, "END")
(END, "END") (ERRORCHAR, ".")
```

- 上图是实验 1 的实验结果，留意到画横线出的记号，输出的二元组格式是（属性名，属性值），其中属性值是一个字符数组类型，会直接把记号记录出来，因此输出的是”.2”和”0.2”，这些本质上是一个字符数组，并不是一个浮点数。
- 下图是实验 2 的实验结果，留意到同样的地方，输出的都是”0.200000”，是默认浮点数输出精度，正是因为把 yytext 字符串的内容转化为浮点数，并且把输出函数改成以”%f”形式输出。
- 其余的 id 和整形数也是，只不过通过两图的对比无法得出而已。



四、心得与体会

1. 进一步掌握 flex 词法分析器生成工具的使用和 flex 与 gcc 的联用。
2. 学会在 lex 源程序中的第三部分使用辅助函数，如本实验中的三个辅助函数分别是对词法单元中的整型数，浮点数和标识符进行转换，转换成对应的变量而不是 yytext 的字符串变量，实际上这才是他们的真正的属性值。（例如整型数的属性值是这个数本身而不是这个数本身的字符串，而如果是默认情况下，在进行词法单元识别时会把这个整型数当成字符串处理，这是不符合需求的。）
3. 由于没有和 Yacc 连用，所以实际上在这里主函数也是辅助函数的一部分。第三部分的辅助函数是用 c 语言写的，用于对词法分析的辅助，如输入输出。
4. 理解了用 C 语言写的辅助函数对于词法分析的重要作用。这些辅助函数一方面可以作为语义动作的一部分，另一方面可以为词法分析提供依据。而且可以将 main 函数作为辅助函数放在这里，当单独使用词法分析器的时候可以在 main 函数里实现对词法分析器的调度和使用。
5. 动作中有返回语句时，返回值应当事先在定义段的第一部分用 C 语言的语法进行定义，以便于后面的操作和使用。
6. 了解了 yylval 这一变量，它的类型与属性值栈（Yacc 中）元素的类型相同，在这里用于存储属性值，把它定义为联合体类型就能根据实际的词法单元来选择对应的类型存储。