# 将 float 关键字替换成 double 关键字

## 一、实验目的：

通过实验明白词法分析的作用不仅仅在于写词法分析器，它还有很多其他用途。

## 二、实验内容：

写一个 lex 程序，它读入一个文件，将该文件中的所有的单独或连续的一段空白（包括一个或多个空格、制表、换行组成的空白）都替换成一个空格。

## 三、实验要求：

输入为一个文本文件；输出为新的文本文件，该文件在原文本文件的基础上将单独或连续的空白变成一个空格。

在 cygwin 下用 flex 和 gcc 工具将实验调试通过，并写出测试例测试正确性。

## 四、具体实现

### 对实验二的 exam2.l 文件修改如下：

1. 新增对空格（单个）的声明：

```
27  #define STRING      29
28  #define SPACE       30
```

2. 修改空白的翻译规则：

```
59  <INITIAL>{ws}           {return (SPACE);}
60  <INITIAL>while          {return (WHILE);}
61  <INITIAL>do             {return (DO);}
62  <INITIAL>if             {return (IF);}
63  <INITIAL>else           {return (ELSE);}
64  <INITIAL>{id}           {return (ID);}
65  <INITIAL>{number}       {return (NUMBER);}
66  <INITIAL>{string}       {return (STRING);}
```

原先对 ws（对空格、缩进、回车的正规定义）的翻译规则是什么都不做，但该实验要求规定若是一片空白则输出成一个空格，所以要对翻译规则进行修改，返回 SPACE 使输出函数做出对应的动作。

3. 修改输出函数 writeout()：

```
97  FILE* writeout(int c, FILE *newout){
98    switch(c){
99      case SPACE: fprintf(newout, " ", yytext);break;
100     case ERRORCHAR: fprintf(newout, "%s", yytext);break;
101     case RELOP: fprintf(newout, "%s", yytext);break;
102     case WHILE: fprintf(newout, "%s", yytext);break;
103     case DO: fprintf(newout, "%s", yytext);break;
104     case IF: fprintf(newout, "%s", yytext);break;
105     case ELSE: fprintf(newout, "%s", yytext);break;
106     case NUMBER: fprintf(newout, "%s", yytext);break;
107     case ID: fprintf(newout, "%s", yytext);break;
108     case NEWLINE: fprintf(newout, "\n", yytext);break;
109     case COPY: fprintf(newout, "%s", yytext);break;
110     case SEMICOLON: fprintf(newout, "%s", yytext);break;
111     case BRACKET: fprintf(newout, "%s", yytext);break;
112     case OP: fprintf(newout, "%s", yytext);break;
113     case STRING: fprintf(newout, "%s", yytext);break;
114     default:break;
115   }
116   return newout;
117 }
```

- ✓ 以往的 writeout 函数传入的参数只有一个，即记号对应的声明（int），因为实验要求此时要将输入文件的内容去掉大段空白后复制到新的文件中，因此再传入一个参数，该参数是这个文件指针 FILE *newout，指向输出的文件。
- ✓ 因此 fprintf 函数的第一个参数从之前的 yyout（Lex 中本身已定义的输出文件指针，该变量指明了 Lex 生成的词法分析器输出到哪里，默认是屏幕输出）改成传参文件指针；
- ✓ 同时输出内容仅仅是记号本身，而不是（记号，属性值）。
- ✓ 新增对正规定义 ws 所识别出的记号进行打印，按照要求，打印一个空格即可。
- ✓ 函数此时有返回值，返回文件指针，以记录此时记录的文件指针位置，方便下次继续写入。

4. 测试文件 test.p 新增内容如下：

```
1
2   //这是第一条注释，双斜杠的
3
4   do {
5       if((i * 0.5) != (j / 2))
6           _a = b_2 + 1.2E-2;
7       else
8           b_2 = _a - 1;
9   }while (c <= 2);
10
11
12
13  /*
14  这是
15  第二条注释
16  斜杠星号的
17  */
18
19
20
21
22
23
24
```

测试文件中包含：

- ✓ 第 5 行后是一行空格。
- ✓ 第 11、19~23 行是 5 个制表符。
- ✓ 第 1、310、12、18、24 行是一行空格。
- ✓ 综上第 10~12 和 18~24 各组成一段空白，第 5 行后也是一段空白，这些应当成为一个空格。

5. 对主函数的修改

```
120 int main (int argc, char ** argv){
121     int c;
122     if (argc>=2){
123       if ((yyin = fopen(argv[1], "r")) == NULL){
124         printf("Can't open file %s\n", argv[1]);
125         return 1;
126       }
127       if (argc>=3){
128         yyout=fopen(argv[2], "w");
129       }
130     }
131
132     FILE *newout;
133     if((newout = fopen("newout.p", "w")) == NULL){
134         printf("Can't open file %s\n", "newout.p");
135         return 1;
136     }
137
138     while (c = yylex())
139         newout = writeout(c, newout);
140
141     if(argc>=2){
142       fclose(yyin);
143       if (argc>=3) fclose(newout);
144     }
145     return 0;
146 }
```

- 申明一个文件指针，并以写入方式（w）在根目录打开“newout.p”文件，即输出文件，若没有则创建。
- 传参时把该文件指针传过去并对指针（指向当前文件内容的位置），并通过函数进行更新，使得记号能正确（顺序）写入输入文件。
- 使用完后关闭文件。

6. 在 cygwin 下用 flex 编译器、gcc 编译器编译得到 a.exe 文件，用测试文件 test.p 进行测试。根目录下生成一个“newout.p”文件，文件内容如下：

```
1   do { if((i * 0.5) != (j / 2)) _a = b_2 + 1.2E-2; else b_2 = _a - 1; }while (c <= 2);
```

只有一行，并且没有大段空白。
经核对，结果正确。

## 五、 心得与体会

1. 实验比较简单，只要清楚 Lex 源程序中的 yyout 的意思即可，或者说知道 Lex 中默认的输出文件指针即可，通过对该指针的修改即能完成实验。
2. 回顾了 C 语言中的文件操作。
3. 词法分析不仅仅能够用来写词法分析器，而且可以实现输入文件的格式转换，字符串替换等其他操作。
4. 每次识别一串空白后就将一个空格写入输出文件，识别到其他内容就原样写入输出文件，因此应有能够区分空白和其他字符串的正规式，原有的正规定义就能符合要求。

## 一、实验内容：

写一个 lex 程序，它读入一个 c 语言文件，将其中所有的 float 关键字都替换成 double 关键字。

## 二、实验要求：

输入为一个 C 语言源文件；输出为新的 C 语言源文件，该文件在原输入的基础上将 float 关键字替换成 double 关键字。

注意：必须是 float 关键字，如果是 afloat 或者 floata 这样的 id 不可以被替换为 adouble 和 doublea。

在 cygwin 下用 flex 和 gcc 工具将实验调试通过，并写出测试例测试正确性。

## 三、具体实现

### 对实验三的 exam2.1 文件修改如下：

1. 新增对浮点数和双精度浮点数关键字的声明：

```
11  #define DOUBLE      7
12  #define FLOAT       8
```

2. 新增对关键字 double 和 float 的翻译规则：

```
67  <INITIAL>double         {return (DOUBLE);}
68  <INITIAL>float          {return (FLOAT);}
```

3. 修改输出函数 writeout()：

```
102 FILE* writeout(int c, FILE *newout){
103   switch(c){
104     case SPACE: fprintf(newout, "%s", yytext);break;
105     case ERRORCHAR: fprintf(newout, "%s", yytext);break;
106     case RELOP: fprintf(newout, "%s", yytext);break;
107     case WHILE: fprintf(newout, "%s", yytext);break;
108     case DO: fprintf(newout, "%s", yytext);break;
109     case IF: fprintf(newout, "%s", yytext);break;
110     case FLOAT: fprintf(newout, "%s", "double");break;
111     case DOUBLE: fprintf(newout, "%s", yytext);break;
112     case ELSE: fprintf(newout, "%s", yytext);break;
113     case NUMBER: fprintf(newout, "%s", yytext);break;
114     case ID: fprintf(newout, "%s", yytext);break;
115     case NEWLINE: fprintf(newout, "\n", yytext);break;
116     case COPY: fprintf(newout, "%s", yytext);break;
117     case SEMICOLON: fprintf(newout, "%s", yytext);break;
118     case BRACKET: fprintf(newout, "%s", yytext);break;
119     case OP: fprintf(newout, "%s", yytext);break;
120     case STRING: fprintf(newout, "%s", yytext);break;
121     default:break;
122   }
123   return newout;
124 }
```

- ✓ 对匹配到正规定义的 ws 的记号应当直接输出，而不是和实验三一样用一个空格代替。
- ✓ 对匹配到关键字 float 的关键字的用 double 代替，由于是最长匹配原则，所以对于 floata 这种记号 Lex 不会识别成 float 关键字，所以不会被替代。

4. 修改测试文件 test.p 内容如下：

```
1    float _a;
2    double b_2;
3    double afloat;
4
5    do {
6        if((i * 0.5) != (afloat / 2))
7            _a = b_2 + 1.2E-2;
8        else
9            b_2 = _a - 1;
10   }while (c <= 2);
11
```

测试文件中包含：
- ✓ 关键字 double 和 float。
- ✓ 带有“float”串的变量名。

5. 对主函数的修改
   文件名后缀改成.c 即可。

6. 在 cygwin 下用 flex 编译器、gcc 编译器编译得到 a.exe 文件，用测试文件 test.c 进行测试。根目录下生成一个“newout.c”文件，文件内容如下：

```
1    double _a;
2    double b_2;
3    double afloat;
4
5    do {
6        if((i * 0.5) != (afloat / 2))
7            _a = b_2 + 1.2E-2;
8        else
9            b_2 = _a - 1;
10   }while (c <= 2);
11
```

- ✓ 第 1 行 float 关键字改成 double 关键字。
- ✓ 第 3 行和第 6 行的 afloat 变量名没有变成 adouble。
- ✓ 其他内容，包括空白和回车未经过修改。
- ✓ 经核对，结果正确。

## 四、 心得与体会

1. 可以利用词法分析对源程序文件进行修改，比如关键字的替换等。
2. 由于 Lex 源程序对记号的匹配是尽可能长的匹配规则，所以不需要担心带有 float 串的记号或字符串会被匹配成关键字并对其改成 double。
3. 在词法规则段，每当识别到一个“float”关键字就向输出文件中写入一个“double”串，识别到空白或其他字符串就原样写入输出文件。
4. 进一步了解了 Lex 源程序的使用。