词法分析与语法分析

本实验任务是编写一个程序对使用CMINUS语言书写的源代码进行词法分析和语法分析（CMINUS语言的文法参见附录），并打印分析结果。实验要求使用词法分析工具GNU Flex和语法分析工具GNU Bison，并使用C语言来完成。虽然版本不一样，但GNU Flex, Bison的使用方法与课本上介绍的Lex, Yacc基本相同。

Flex的前身是Lex。Lex是1975年由Mike Lesk和当时还在AT&T做暑期实习的Eric Schmidt，共同完成的一款基于Unix环境的词法分析程序生成工具。虽然Lex很出名并被广泛使用，但它的低效和诸多问题也使其颇受诟病。后来伯克利实验室的Vern Paxson使用C语言重写Lex，并将这个新的程序命名为Flex（意为Fast Lexical Analyzer Generator）。无论在效率上还是在稳定性上，Flex都远远好于它的前辈Lex。我们在Linux下使用的是Flex在GNU License下的版本，称作GNU Flex。 GNU Flex在Linux下的安装非常简单。你可以去它的官方网站上下载安装包自行安装，不过在基于Debian的Linux系统下，更简单的安装方法是直接在命令行敲入如下命令：

```
sudo apt-get install flex
```

Bison的前身为基于Unix的Yacc。令人惊讶的是，Yacc的发布时间甚至比Lex还要早。Yacc所采用的LR分析技术的理论基础早在50年代就已经由Knuth逐步建立了起来，而Yacc本身则是贝尔实验室的S.C. Johnson基于这些理论在75年到78年写成的。到了1985年，当时在UC Berkeley的一个研究生Bob Corbett在BSD下重写了Yacc，后来GNU Project接管了这个项目，为其增加了许多新的特性，于是就有了我们今天所用的GNU Bison。 GNU Bison在Linux下的安装非常简单，你可以去它的官方网站上下载安装包自行安装，基于Debian的Linux系统下更简单的方法同样是直接在命令行敲入如下命令：

```
sudo apt-get install bison
```

参考：《flex & bision》，John Levine著，陆军译，东南大学出版社， 2011年。

1. 实验内容

1.1 实验要求

你的程序要能够查出CMINUS源代码中可能包含的下述几类错误：

1) **词法错误（错误类型A）**：即出现CMINUS词法中未定义的字符以及任何不符合CMINUS词法单元定义的字符；
2) **语法错误（错误类型B）**。

除此之外，你的程序可以选择完成以下部分或全部的要求：

1) **要求1.1**：识别八进制数和十六进制数。若输入文件中包含符合词法定义的八进制数（如0123）和十六进制数（如0x3F），你的程序需要得出相应的词法单元；若输入文件中包含不符合词法定义的八进制数（如09）和十六进制数（如0x1G），你的程序需要给出输入文件有词法错误（即错误类型A）的提示信息。八进制数和十六进制数的定义参见附录A。
2) **要求1.2**：识别指数形式的浮点数。若输入文件中包含符合词法定义的指数形式的浮点数（如1.05e-4），你的程序需要得出相应的词法单元；若输入文件中包含不符合词法定义的指数形式的浮点数（如1.05e），你的程序需要给出输入文件有词法错误（即错误类型A）的提示信息。指数形式的浮点数的定义参见附录A。
3) **要求1.3**：识别“//”和“/*…*/”形式的注释。若输入文件中包含符合定义的“//”和“/*…*/”形式的注释，你的程序需要能够滤除这样的注释；若输入文件中包含不符合定义的注释（如“/*…*/”注释中缺少“/*”），你的程序需要给出由不符合定义的注释所引发的错误的提示信息。注释的定义参见附录A。

你的程序在输出错误提示信息时，需要输出具体的错误类型、出错的位置（源程序行号）以及相关的说明文字。

1.2 输入格式

你的程序的输入是一个包含CMINUS源代码的文本文件，程序需要能够接收一个输入文件名作为参数。例如，假设你的程序名为cc、输入文件名为test1、程序和输入文件都位于当前目录下，那么在Linux命令行下运行./cc test1即可获得以test1作为输入文件的输出结果。

1.3 输出格式

实验一要求通过标准输出打印程序的运行结果。对于那些包含词法或者语法错误的输入文件，只要输出相关的词法或语法有误的信息即可。在这种情况下，注意不要输出任何与语法树有关的内容。要求输出的信息包括错误类型、出错的行号以及说明文字，其格式为：

Error type [错误类型] at Line [行号]: [说明文字].

说明文字的内容没有具体要求，但是错误类型和出错的行号一定要正确，因为这是判断输出的错误提示信息是否正确的唯一标准。请严格遵守实验要求中给定的错误分类（即词法错误

为错误类型A，语法错误为错误类型B），否则将影响你的实验评分。注意，输入文件中可能会包含一个或者多个错误（但输入文件的同一行中保证不出现多个错误），你的程序需要将这些错误全部报告出来，每一条错误提示信息在输出中单独占一行。

对于那些没有任何词法或语法错误的输入文件，你的程序需要将构造好的语法树按照先序遍历的方式打印每一个结点的信息，这些信息包括：

1) 如果当前结点是一个语法单元并且该语法单元没有产生 ，则打印该语法单元的名称以及它在输入文件中的行号（行号被括号所包围， 并且与语法单元名之间有一个空格）。所谓某个语法单元在输入文件中的行号是指该语法单元产生出的所有词素中的第一个在输入文件中出现的行号。
2) 如果当前结点是一个语法单元并且该语法单元产生了 ，则无需打印该语法单元的信息。
3) 如果当前结点是一个词法单元，则只要打印该词法单元的名称，而无需打印该词法单元的行号。
    a) 如果当前结点是词法单元ID，则要求额外打印该标识符所对应的词素；
    b) 如果当前结点是词法单元TYPE，则要求额外打印说明以该类型为int还是float；
    c) 如果当前结点是词法单元INT或者FLOAT，则要求以十进制的形式额外打印该数字所对应的数值；
    d) 词法单元所额外打印的信息与词法单元名之间以一个冒号和一个空格隔开。

每一条词法或语法单元的信息单独占一行，而每个子结点的信息相对于其父结点的信息来说，在行首都要求缩进2个空格。具体输出格式可参见后续的样例。

2. 测试环境

你的程序将在如下环境中被编译并运行：

1) GNU Linux Release: Ubuntu 12.04, kernel version 3.2.0-29；
2) GCC version 4.6.3；
3) GNU Flex version 2.5.35；
4) GNU Bison version 2.5。

一般而言，只要避免使用过于冷门的特性，使用其它版本的Linux或者GCC等，也基本上不会出现兼容性方面的问题。注意，实验一的检查过程中不会去安装或尝试引用各类方便编程的函数库（如glib等），因此请不要在你的程序中使用它们。

3. 提交要求

实验一要求提交如下内容：

1) Flex、Bison以及C语言的可被正确编译运行的源程序。
2) 一份PDF格式的实验报告，内容包括：

a) 你的程序实现了哪些功能？简要说明如何实现这些功能。清晰的说明有助于助教对你的程序所实现的功能进行合理的测试。

b) 你的程序应该如何被编译？可以使用脚本、makefile或逐条输入命令进行编译，请详细说明应该如何编译你的程序。无法顺利编译将导致助教无法对你的程序所实现的功能进行任何测试，从而丢失相应的分数。

c) 实验报告的长度不得超过三页！所以实验报告中需要重点描述的是你的程序中的亮点，是你认为最个性化、最具独创性的内容，而相对简单的、任何人都可以做的内容则可不提或简单地提一下，尤其要避免大段地向报告里贴代码。实验报告中所出现的最小字号不得小于五号字（或英文11号字）。

4. 测试样例（必做内容）

助教将使用下述测试样例来测试你的程序，请仔细阅读样例，以加深对实验要求以及输出格式要求的理解，使得你所提交程序的输出满足既定要求。

内容样例。

样例1：

输入（行号是为标识需要，并非样例输入的一部分，后同）：

```
1 int main() 2  {
3       int i = 1;
4       int j = ~i;
5  }
```

输出：

这个程序存在词法错误。第4行中的字符“~”没有在我们的CMINUS词法中被定义过，因此你的程序可以输出如下的错误提示信息：

```
Error type A at Line 4: Mysterious character "~".
```

样例2：

输入：

```
1 int main() 2  {
3       float a[10][2];
4       int i;
5       a[5,3] = 1.5;
6       if (a[1][2] == 0) i = 1 else i = 0;
7  }
```

输出：

这个程序存在两处语法错误。其一，虽然我们的程序中允许出现方括号与逗号等字符，但二维数组正确的访问方式应该是a[5][3]而非a[5,3]。其二，第6行的if-else语句在else之前少了一个分号。因此你的程序可以输出如下的两行错误提示信息：

```
Error type B at Line 5: Missing "]". Error type B at
Line 6: Missing ";".
```

样例3：

输入：

```
1 int inc() 2  {
3       int i;
4       i = i + 1;
5  }
```

输出：

这个程序非常简单，也没有任何词法或语法错误，因此你的程序需要输出如下的语法树结点信息（行号是为标识需要，并非程序输出的一部分，后同）：

```
1  Program (1)
2    ExtDefList (1)
3      ExtDef (1)
4        Specifier (1)
5          TYPE: int
6        FunDec (1)
7          ID: inc
8          LP
9          RP
10       CompSt (2)
11         LC
12         DefList (3)
13           Def (3)
14             Specifier (3)
15               TYPE: int
16             DecList (3)
17               Dec (3)
18                 VarDec (3)
19                   ID: i
20             SEMI
21         StmtList (4)
22           Stmt (4)
23             Exp (4)
24               Exp (4)
25               ID: i
26               ASSIGNOP
27               Exp (4)
28                 Exp (4)
29                   ID: i
30               PLUS
31               Exp (4)
32                 INT: 1
33             SEMI
34         RC
```