实验一 词法分析和语法分析

组长：褚有刚（学号：171860526）

组员：夏宇（学号：171180554）

一、实验的目的

利用*GUN Flex*以及*GUN Bison*两个工具，编写一个对使用C语言写成的源代码进行词法分析和语法分析的简单工具，以加强对词法分析以及语法分析的认识和了解。

二、实验的流程

1. 设计正则表达式
   正则表达式按照实验指导书的附录部分的规约设计即可；
2. 设计文法
   文法在实验指导书的附录部分已经给出，需要调整部分产生式的优先级以消除冲突：
   1. 悬挂else

      通过申明一个额外的优先级低于`ELSE`的标记：`LOWER_THAN_ELSE`，将其优先级赋予给`stmt -> if lp exp rp stmt`以保证其优先级低于`stmt -> if lp exp rp stmt else stmt`来消除移入/归约冲突；
   2. 表达式
      C--语法中的运算符的优先级以及结合性已经在实验指导书的附录中给出，需要注意的是对于'-'，如果其为减法，则其优先级与结合性和'+'相同，但是如果其为负号，则其优先级需要提高，需要定义一个额外和运算符`NOT`优先级以及结合性相同的标记`UMINUS`，使用`%prec`将其优先级赋予`exp -> - exp`产生式；
3. 设计语法树

- 语法树是一棵多叉树，在实现中我们使用二叉树表示多叉树数据结构，对于二叉树中每个节点，其两个子节点分别为该节点的`firstChild`和`nextSibling`；
- 在我们的设计中，语法树的每个节点都代表一个文法符号，对于词法单元，在词法分析器识别了一个词法单元的时候创建并记录其语义值然后返回给语法分析器，语法分析器归约的时候将其加入到语法树中；对于语法单元，在归约的时候，创建代表产生式左边的非终结符的节点，并将其右边的文法符号串作为子节点加入，对于开始符号，还需要将其语义值赋值给语法树的根`root`；
- 在语法树的每个节点中记录了输出语法树时需要的信息：文法符号类型，行号，以及其他额外的信息（见节点的定义），
- 在创建好语法树之后，如果没有发生错误，只需要按照先序遍历的顺序输出树中的节点，在输出的过程中通过局部变量记录语法树的层次以产生合理的缩进即可。

4. 错误处理和恢复
   1. 词法分析

设计特定的模式匹配常见的出错情形，例如：八/十六进制数字，注释，数字开头的标识符，为了防止因为词法错误引起的语法错误，在必要的时候返回预测的词法单元；对于其他的错误，报错不返回；

2. 语法分析

根据编者自己的编程经验，对于常见的语法错误，在文法中添加包含了`error`标记的产生式，为了使得报错尽可能的准确，尽量添加在较底层的产生式中，并且尽可能使用精确的错误匹配模式，由于发生错误的时候调用的`yyerror`函数可定制性太差，我们在该函数中仅仅是对错误进行了计数并且记录了错误的位置信息，当使用错误产生式进行归约的时候使用自定义的错误输出函数来输出错误信息。

三、实验中的遇到的问题和解决方案

1. 语言不支持的特性
   对于我们组的C--语言不支持的特性（注释，八/十六进制数字的识别），当用户在尝试使用这些特性的时候，会输出大片无关的错误信息从而引起用户的恐慌，因此我们仍然在词法分析器中添加了相应的正则表达式进行匹配，输出编译器不支持这些特性的类似的提示信息并且给语法分析器返回了正确的词法单元以保证不会因此而引起大片的语法错误；
2. 更友好的提示信息
   在C语言的编译器中，在检测到错误的同时会输出发生错误的那行代码并且指出列号，因此在我们的编译器中也部分实现了这个功能，在词法分析中，当遇到'\n'时，我们的词法分析器会自动缓存下一行到全局变量`linebuf`中，以期待当发生错误的时候将该行输出，但是由于错误可能在距离发生点多行之后才能够被检测到，但是为了不加重词法分析器的负担，我们只在错误检测点和错误发生点在同一行的时候进行输出；
3. 丢失的同步符号
   为了输出准确的提示信息，我们在文法中添加了较多的错误产生式并且尽可能添加在了底层的位置，但是这样带来的问题是可能在发生语法错误时一直没有找到同步符号而没有输出任何提示信息而退出编译程序，为了避免这种情况发生，使用了一个全局变量记录了上次发生错误的位置，当发生错误但是没有输出信息的时候输出该位置并提示发生了语法错误；
4. 内存泄漏的处理
   当语法分析器在发生错误的时候，会弹出栈中的若干非终结符，而这些非终结符的语义值中具有额外的内存申请记录，因此如果不进行释放会造成内存泄漏，为了防止这种情况，使用了`bison`的特性中的`%destructor`进行了内存的释放；