编译原理

课程Project说明

COMP130014.01

2022.09

简介

- 本学期共有两个Project：
- Project 1：词法分析，占比**40%**
- Project 2：语法分析，占比**60%**
- 每组1-2人，**2022年10月7日23:59**之前确定组员名单发送至负责Project的TA邮箱（标题2022编译课程组队+姓名+学号）
- TA联系方式：
  - 何瑞安 rahe21@m.fudan.edu.cn (负责Project)
  - 谈天 tant21@m.fudan.edu.cn (负责课程作业)
- 面对面建群，发布Project/答疑/预约汇报时间

实验环境

- OS：Linux
- 依赖：gcc/g++, make，flex, bison
- gcc，make，flex与bison安装(以Ubuntu为例)：
  - sudo apt-get install build-essential flex bison
- 实验环境也可自行在MAC OS以及WINDOWS下配置，建议使用虚拟机安装Ubuntu。
- 简单来说，就是C/C++配合flex与bison两个工具完成实验

实验目的

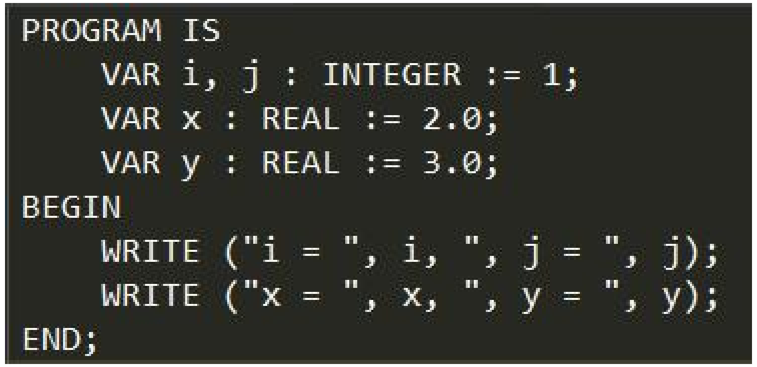
- 通过flex与bison，分析目标PCAT语言，并**生成目标语言的语法树**
- PCAT语言可看作一种简化版的PASCAL语言：

```
PROGRAM IS
    VAR i, j : INTEGER := 1;
    VAR x : REAL := 2.0;
    VAR y : REAL := 3.0;
BEGIN
    WRITE ("i = ", i, ", j = ", j);
    WRITE ("x = ", x, ", y = ", y);
END;
```

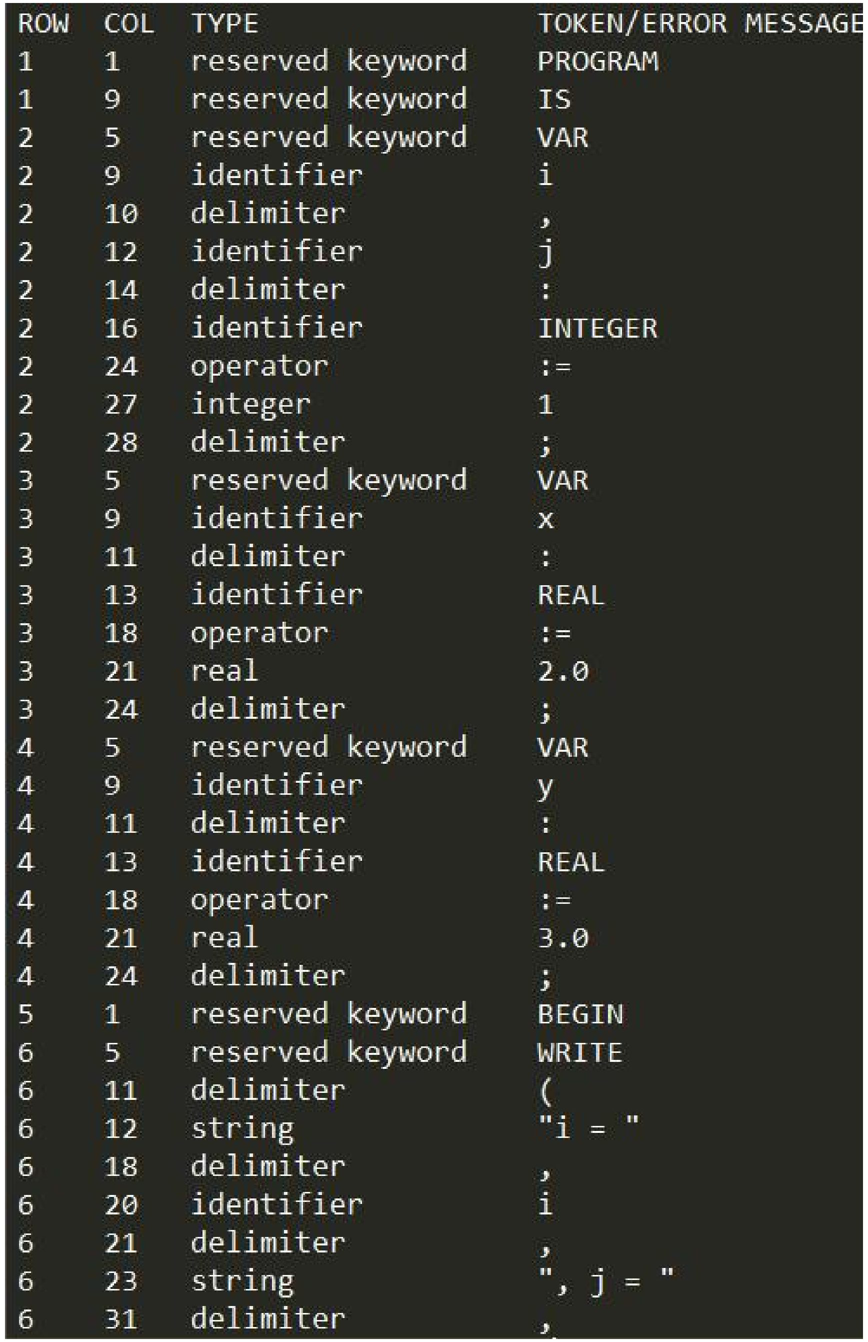
Project 1 词法分析 (40%)

- 任务：使用flex工具对于给定的PCAT语言样例做词法分析

- 左侧：PCAT语言代码
- 右侧：词法分析的结果



```
PROGRAM IS
    VAR i, j : INTEGER := 1;
    VAR x : REAL := 2.0;
    VAR y : REAL := 3.0;
BEGIN
    WRITE ("i = ", i, ", j = ", j);
    WRITE ("x = ", x, ", y = ", y);
END;
```



```
ROW  COL  TYPE                TOKEN/ERROR MESSAGE
1    1    reserved keyword    PROGRAM
1    9    reserved keyword    IS
2    5    reserved keyword    VAR
2    9    identifier          i
2    10   delimiter           ,
2    12   identifier          j
2    14   delimiter           :
2    16   identifier          INTEGER
2    24   operator            :=
2    27   integer             1
2    28   delimiter           ;
3    5    reserved keyword    VAR
3    9    identifier          x
3    11   delimiter           :
3    13   identifier          REAL
3    18   operator            :=
3    21   real                2.0
3    24   delimiter           ;
4    5    reserved keyword    VAR
4    9    identifier          y
4    11   delimiter           :
4    13   identifier          REAL
4    18   operator            :=
4    21   real                3.0
4    24   delimiter           ;
5    1    reserved keyword    BEGIN
6    5    reserved keyword    WRITE
6    11   delimiter           (
6    12   string              "i = "
6    18   delimiter           ,
6    20   identifier          i
6    21   delimiter           ,
6    23   string              ", j = "
6    31   delimiter           ,
```

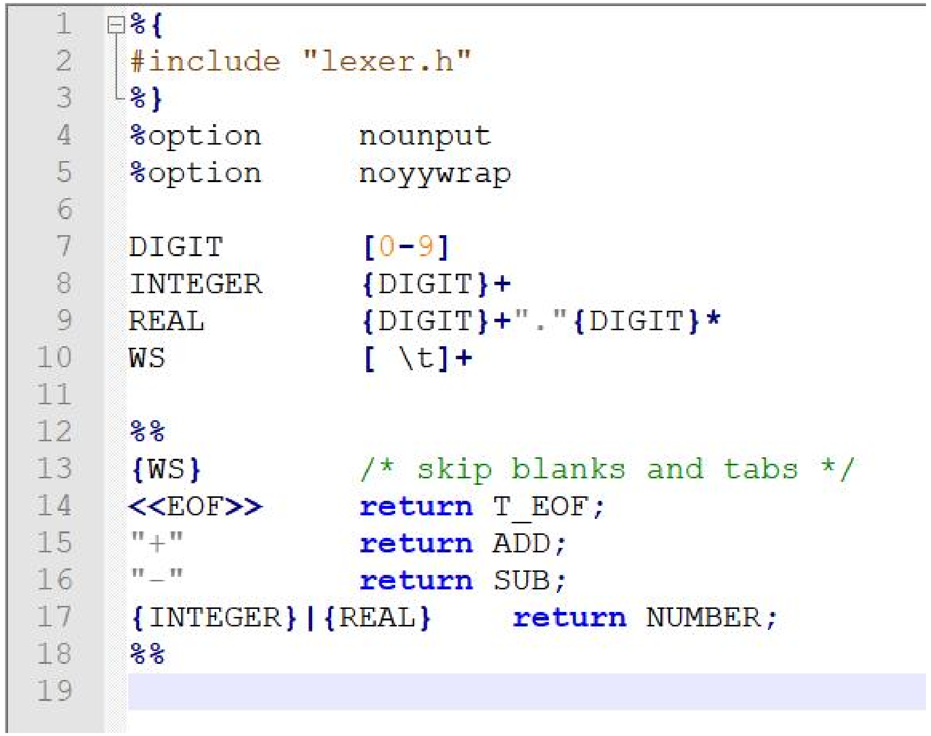
Flex简介

- 一种可以使用正则表达式完成文本词法分析的工具，将正则表达式描述（.lex）转化为C语言解析程序（.c）。

- 举例：提取出只有加法和减法的表达式的token

Demo： 简单的加减法分析器

lexer.lex



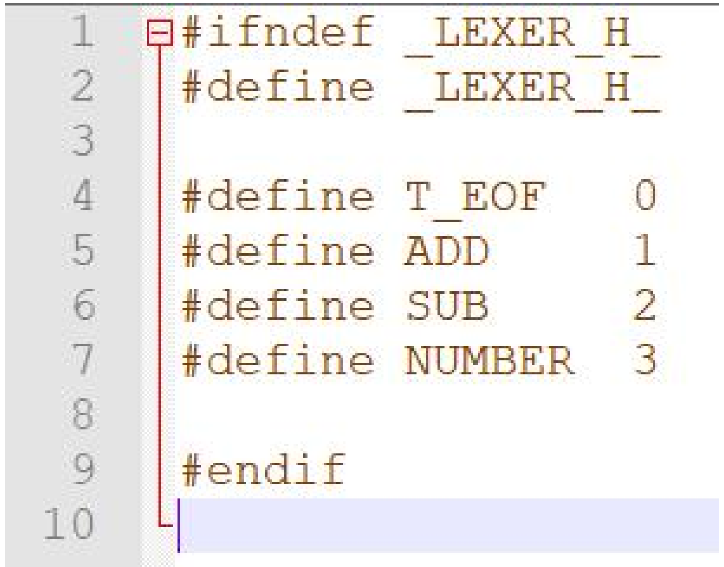
```
%{
#include "lexer.h"
%}
%option     nounput
%option     noyywrap

DIGIT       [0-9]
INTEGER     {DIGIT}+
REAL        {DIGIT}+"."{DIGIT}*
WS          [ \t]+

%%
{WS}        /* skip blanks and tabs */
<<EOF>>     return T_EOF;
"+"         return ADD;
"-"         return SUB;
{INTEGER}|{REAL}    return NUMBER;
%%

```

定义区

规则区

用户代码区

lexer.h



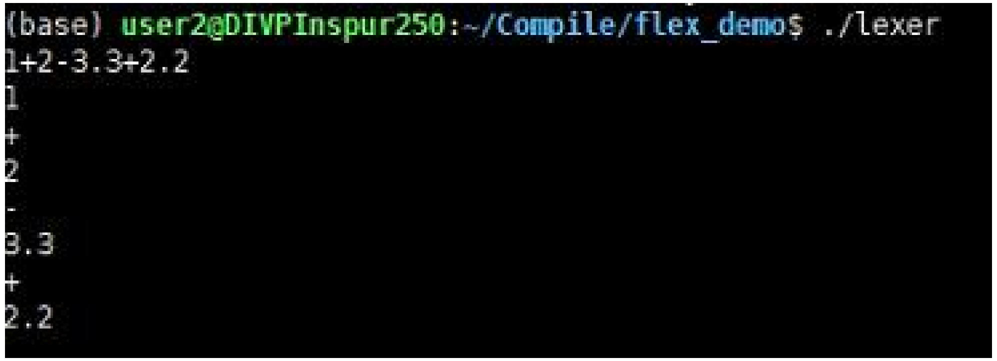
```
#ifndef _LEXER_H_
#define _LEXER_H_

#define T_EOF   0
#define ADD     1
#define SUB     2
#define NUMBER  3

#endif

```

简单的加减法分析器

请仔细观察lexer.c生成规律并阅读flex文档

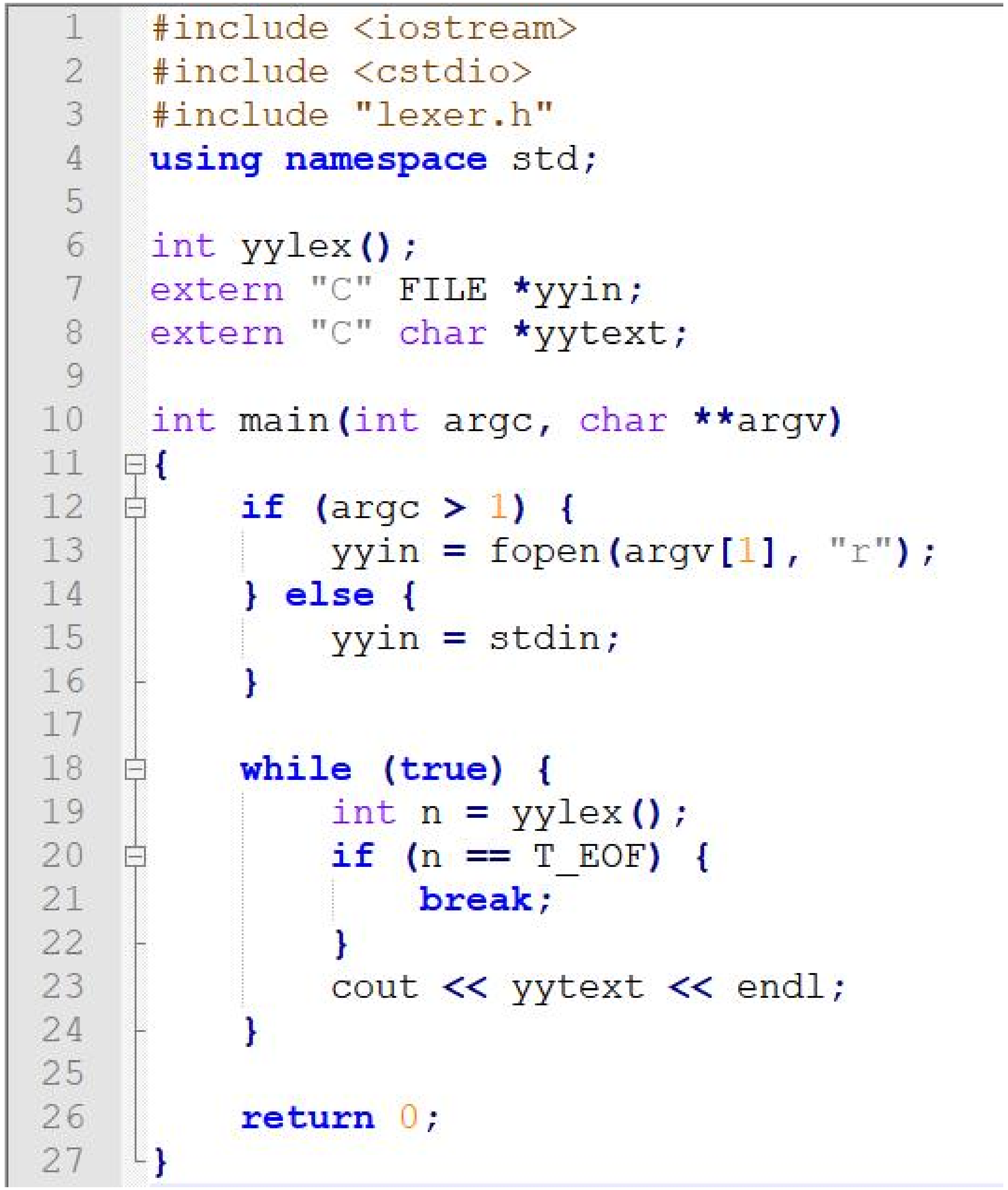
编译:

```
flex -o lexer.c lexer.lex
g++ -c lexer.c -o lexer.o
g++ main.cpp lexer.o –o lexer
```



```
(base) user2@DIVPInspur250:~/Compile/flex_demo$ ./lexer
1+2-3.3+2.2
1
+
2
-
3.3
+
2.2
```

main.cpp



```cpp
#include <iostream>
#include <cstdio>
#include "lexer.h"
using namespace std;

int yylex();
extern "C" FILE *yyin;
extern "C" char *yytext;

int main(int argc, char **argv)
{
    if (argc > 1) {
        yyin = fopen(argv[1], "r");
    } else {
        yyin = stdin;
    }

    while (true) {
        int n = yylex();
        if (n == T_EOF) {
            break;
        }
        cout << yytext << endl;
    }

    return 0;
}
```

参考资料

- 下发文件PCAT语言参考PDF中有相应的词法参考，实现以该说明为准。

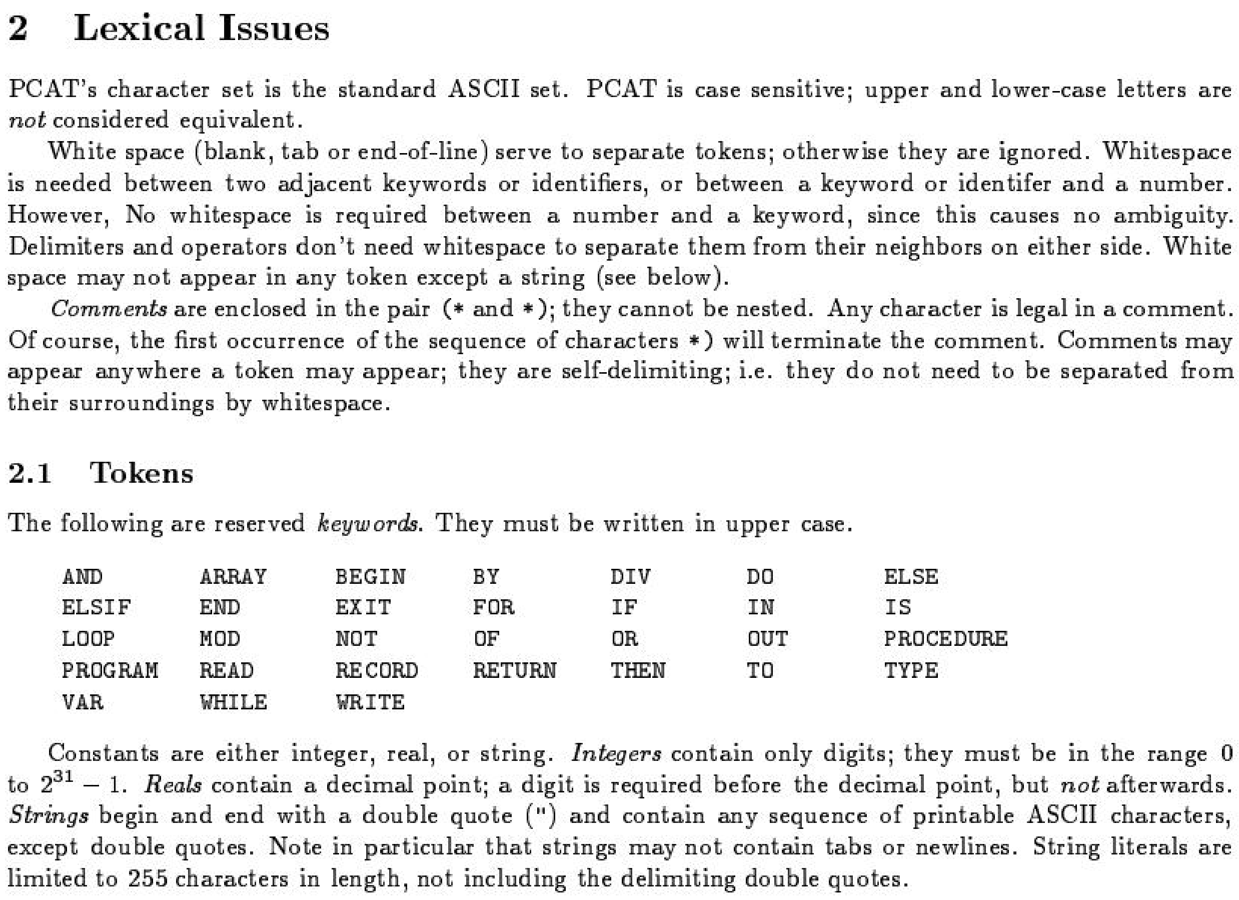
- Flex manual:

http://ranger.uta.edu/~fegaras/cse5317/flex/flex_toc.html

- Bison manual:

http://ranger.uta.edu/~fegaras/cse5317/bison/bison_toc.html

参考资料



2 Lexical Issues

PCAT's character set is the standard ASCII set. PCAT is case sensitive; upper and lower-case letters are *not* considered equivalent.

White space (blank, tab or end-of-line) serve to separate tokens; otherwise they are ignored. Whitespace is needed between two adjacent keywords or identifiers, or between a keyword or identifer and a number. However, No whitespace is required between a number and a keyword, since this causes no ambiguity. Delimiters and operators don't need whitespace to separate them from their neighbors on either side. White space may not appear in any token except a string (see below).

Comments are enclosed in the pair `(*` and `*)`; they cannot be nested. Any character is legal in a comment. Of course, the first occurrence of the sequence of characters `*)` will terminate the comment. Comments may appear anywhere a token may appear; they are self-delimiting; i.e. they do not need to be separated from their surroundings by whitespace.

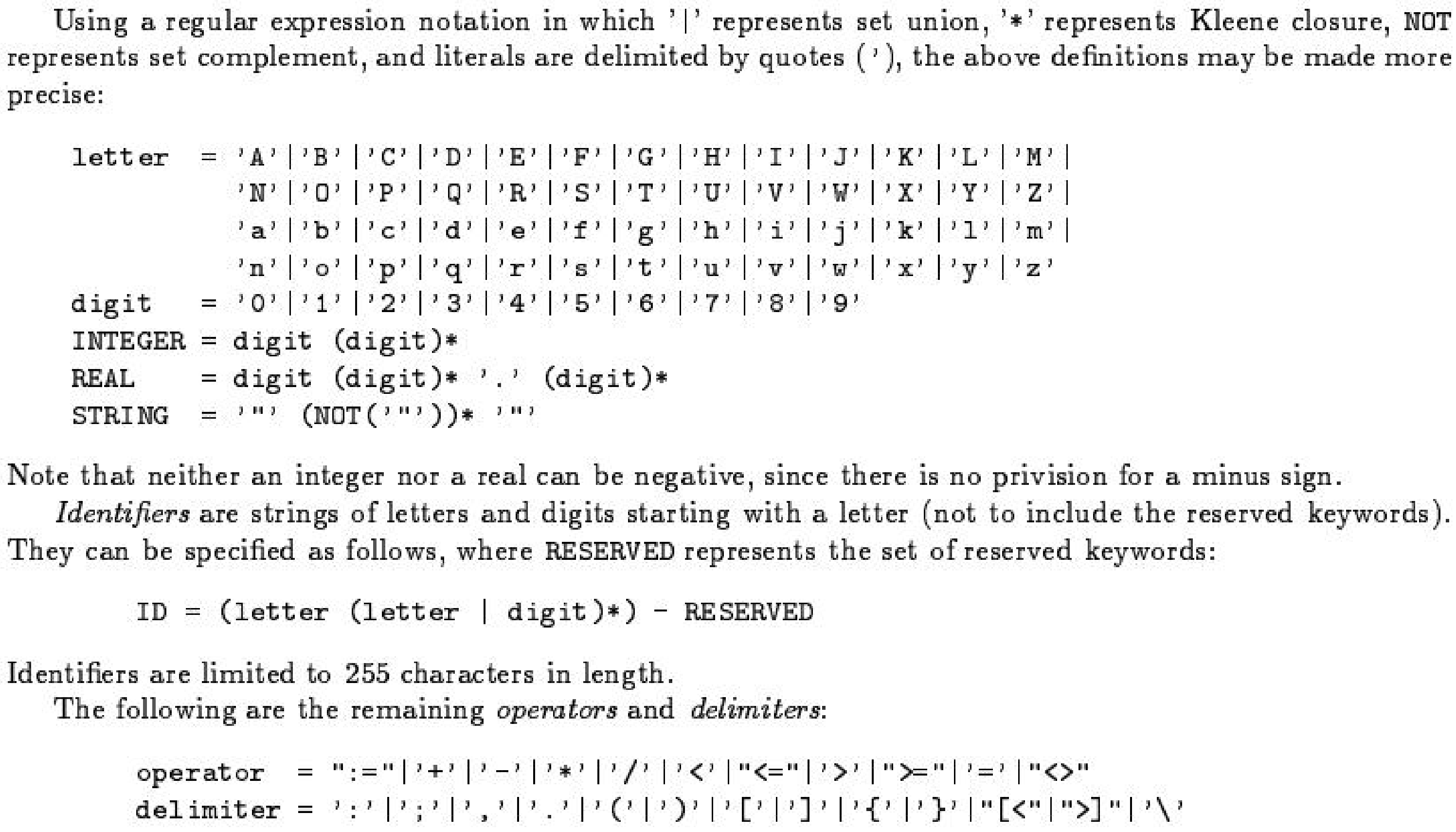
2.1 Tokens

The following are reserved *keywords*. They must be written in upper case.

```
AND      ARRAY    BEGIN    BY       DIV      DO       ELSE
ELSIF    END      EXIT     FOR      IF       IN       IS
LOOP     MOD      NOT      OF       OR       OUT      PROCEDURE
PROGRAM  READ     RECORD   RETURN   THEN     TO       TYPE
VAR      WHILE    WRITE
```

Constants are either integer, real, or string. *Integers* contain only digits; they must be in the range 0 to $2^{31} - 1$. *Reals* contain a decimal point; a digit is required before the decimal point, but *not* afterwards. *Strings* begin and end with a double quote (") and contain any sequence of printable ASCII characters, except double quotes. Note in particular that strings may not contain tabs or newlines. String literals are limited to 255 characters in length, not including the delimiting double quotes.

参考资料



Using a regular expression notation in which ’|’ represents set union, ’*’ represents Kleene closure, NOT represents set complement, and literals are delimited by quotes (’), the above definitions may be made more precise:

```
letter  = ’A’|’B’|’C’|’D’|’E’|’F’|’G’|’H’|’I’|’J’|’K’|’L’|’M’|
          ’N’|’O’|’P’|’Q’|’R’|’S’|’T’|’U’|’V’|’W’|’X’|’Y’|’Z’|
          ’a’|’b’|’c’|’d’|’e’|’f’|’g’|’h’|’i’|’j’|’k’|’l’|’m’|
          ’n’|’o’|’p’|’q’|’r’|’s’|’t’|’u’|’v’|’w’|’x’|’y’|’z’
digit   = ’0’|’1’|’2’|’3’|’4’|’5’|’6’|’7’|’8’|’9’
INTEGER = digit (digit)*
REAL    = digit (digit)* ’.’ (digit)*
STRING  = ’"’ (NOT(’"’))* ’"’
```

Note that neither an integer nor a real can be negative, since there is no privision for a minus sign.

Identifiers are strings of letters and digits starting with a letter (not to include the reserved keywords). They can be specified as follows, where RESERVED represents the set of reserved keywords:

```
ID = (letter (letter | digit)*) - RESERVED
```

Identifiers are limited to 255 characters in length.

The following are the remaining *operators* and *delimiters*:

```
operator  = ":="|’+’|’-’|’*’|’/’|’<’|"<="|’>’|">="|’=’|"<>"
delimiter = ’:’|’;’|’,’|’.’|’(’|’)’|’[’|’]’|’{’|’}’|"[<"|">]"|’\’
```

Project 1 评分细则

项目完成度及正确性：（共计70分）

1. 正确分析case 1-10中出现的所有tokens并统计tokens的总数，将每个case的词法分析结果存储成txt格式**（必做，每个样例完全正确得5分，共计50分）**

2. 正确输出每一个token的起始行号、列号与类型（选做，每个样例完全正确得1分，共计10分）

3. 正确分析case 11中出现的各种无需语法分析的词法错误，提供相应报错信息（选做，25个测试点，其中存在10个错误，共计10分）

项目报告及展示：（共计30分）

1. 撰写项目报告，说明flex的用法，识别不同token所使用的正则表达式及其原理，如何判断token的行列号及类型，如何实现报错功能等等，在结尾标明分工及贡献百分比**（必做，20分）**

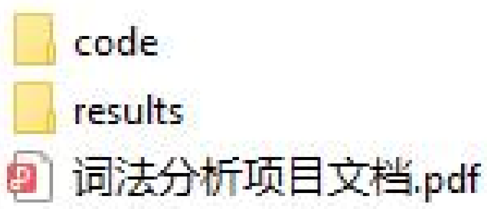
2. 项目报告完成后，与TA预约，在上机课时间向TA展示样例的词法分析结果，TA会就项目相关内容进行简单的提问**（必做，10分）**

Project 1 提交方式

- 项目代码，运行结果（txt）及项目报告（PDF）请打包（zip）并发送至TA邮箱 rahe21@m.fudan.edu.cn
- 邮件/压缩包标题：2022编译原理PJ1 姓名1 姓名2
- **项目报告DDL: 2022年11月4日 23:59**
- 提交报告后与助教预约展示时间
- **展示DDL: 2022年11月11日上机课**
- **Project1讲解与Project2发布：2022年11月11日上机课**

- 如果文件太大，可先上传至百度云或者复旦云，再将网盘分享地址发送到TA邮箱
- 若对Project有疑问，或想在上机课外时间展示，可与TA联系
- TA办公地址：江湾校区交叉学科2号楼A4008室

- **严禁抄袭，包括网络上和同学的代码，一经发现Project作0分处理**
- **只实现必做功能也一定可以顺利通过，不要铤而走险**


code
results
词法分析项目文档.pdf