**电子科技大学 信息与软件工程 学院**

**标 准 实 验 报 告**

**（实验）课程名称编译技术**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：袁佳琪 学 号：2021090909029**

**指导教师：周尔强**

**实验时间：2022.10.21**

**实验地点：信软楼303**

**一、实验室名称：信软 303/304**

**二、实验项目名称：词法分析器**

**三、实验学时：4学时**

**四、实验内容及步骤：**

**实验内容：**

1. 以 test\_cases 中的文件作为输入，编写程序（c/c++）

(1)识别程序中所有的常数、运算符、界符、标识符及关键字

(2)将所编写程序命名为“man\_lex.c”

2. 学习所提供资料中的简单 flex 源程序，试着修改其中的规则及代码，再运行之

3. 用 flex 完成 任务1，并将所编写flex源文件命名为auto\_lex.l

**实验步骤：**

1. 首先编写Flex程序的输入文件。输入文件由声明、定义、规则和用户定义函数四部分构成。声明部分包含了C语言相关声明，如头文件等等；定义部分负责为正则表达式命名，以提高规则部分程序的可读性；规则是最重要的一部分，此部分编写了对每一个所识别符号将要采取的动作；最后用户定义函数的部分可编写用户将实现功能的函数，这些函数将会被复制到词法分析器中。
2. 第二步，通过“flex autoLex.l”命令，将编写的文件作为Flex程序的输入文件，生成词法分析器lex.yy.c文件。
3. 第三步，通过“gcc -o scan lex.yy.c -lfl”命令，将词法分析器文件与main函数与yywrap函数链接后，生成名为“scan”的可执行文件。最后，通过执行scan程序即可进行词法分析。

代码如下：

1. 编写C语言代码，使能识别常数、运算符、界符、标识符、关键字以及无视单行注释，代码如下：

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <ctype.h>  #include <string.h>  #define MAX\_LEN 100  #define LETTER 0  #define DIGIT 1  #define UNKNOWN 999  #define UNDERLINE 2  #define POINT 3  int charClass;  char lexeme[MAX\_LEN];  char nextChar;  char next2Char;  int lexLen;  int token;  int nextToken;  FILE \*f;  enum {  num\_INT = 256,  num\_FLOAT = 257,  Y\_ID = 258,    Y\_FOR = 259,  Y\_FLOAT = 260,  Y\_INT = 261,  Y\_VOID = 262,  Y\_CONST = 263,  Y\_IF = 264,  Y\_ELSE = 265,  Y\_WHILE = 266,  Y\_BREAK = 267,  Y\_CONTINUE = 268,  Y\_RETURN = 269,  Y\_ADD = 270,  Y\_SUB = 271,  Y\_MUL = 272,  Y\_DIV = 273,  Y\_MODULO = 274,  Y\_LESS = 275,  Y\_LESSEQ = 276,  Y\_GREAT = 277,  Y\_GREATEQ = 278,  Y\_NOTEQ = 279,  Y\_EQ = 280,  Y\_NOT = 281,  Y\_AND = 282,  Y\_OR = 283,  Y\_ASSIGN = 284,  Y\_LPAR = 285,  Y\_RPAR = 286,  Y\_LBRACKET = 287,  Y\_RBRACKET = 288,  Y\_LSQUARE = 289,  Y\_RSQUARE = 290,  Y\_COMMA = 291,  Y\_SEMICOLON = 292,  Y\_EOF = 293  };  char \*keywords[]={"for","float","int","void","const","if","else","while","break","continue","return",0};  void addChar(){  if(lexLen <= MAX\_LEN-2){  lexeme[lexLen++]=nextChar;  lexeme[lexLen]=0;  }  else{  printf("lexeme is too long.\n");  }  }  void getChar(){  static int firstRun = 1;  if(firstRun){  nextChar=getc(f);  next2Char=getc(f);  firstRun=0;  }  else{  nextChar=next2Char;  next2Char=getc(f);  }    if(nextChar == EOF){  charClass=EOF;  }  else{  if(isalpha(nextChar))  charClass=LETTER;  else if(isdigit(nextChar))  charClass=DIGIT;  else if(nextChar=='\_')  charClass=UNDERLINE;  else if(nextChar=='.')  charClass=POINT;  else  charClass=UNKNOWN;  }  }  void getNonBlank(){  while(isspace(nextChar))  getChar();  }  int checkSymbol(char ch,char nextCh){  switch(ch){  case'+':  addChar();  nextToken=Y\_ADD;  break;  case'-':  addChar();  nextToken=Y\_SUB;  break;  case'\*':  addChar();  nextToken=Y\_MUL;  break;  case'/':  addChar();  nextToken=Y\_DIV;  if(nextCh=='/'){  while(nextChar!='\n'){  getChar();  addChar();  }  return 0;  }  break;  case'%':  addChar();  nextToken=Y\_MODULO;  break;  case'(':  addChar();  nextToken=Y\_LPAR;  break;  case')':  addChar();  nextToken=Y\_RPAR;  break;  case'[':  addChar();  nextToken=Y\_LSQUARE;  break;  case']':  addChar();  nextToken=Y\_RSQUARE;  break;  case'{':  addChar();  nextToken=Y\_LBRACKET;  break;  case'}':  addChar();  nextToken=Y\_RBRACKET;  break;  case';':  addChar();  nextToken=Y\_SEMICOLON;  break;  case',':  addChar();  nextToken=Y\_COMMA;  break;  case'=':  addChar();  nextToken=Y\_ASSIGN;  if(nextCh=='='){  getChar();  addChar();  nextToken=Y\_EQ;  }  break;  case'>':  addChar();  nextToken=Y\_GREAT;  if(nextCh=='='){  getChar();  addChar();  nextToken=Y\_GREATEQ;  }  break;  case'<':  addChar();  nextToken=Y\_LESS;  if(nextCh=='='){  getChar();  addChar();  nextToken=Y\_LESSEQ;  }  break;  case '!':  addChar();  nextToken=Y\_NOT;  if(nextCh=='='){  getChar();  addChar();  nextToken=Y\_NOTEQ;  }  break;  case EOF:  addChar();  nextToken=Y\_EOF;  break;  default:  printf("unknown character'%c'.\n",ch);  nextToken=-1;  }  return nextToken;  }  void checkKeywords(char\* pword){  int i = 0;  while(keywords[i]!=0){  char\* pkeyword=keywords[i];  if(strcmp(pword,pkeyword)==0){  nextToken=259+i;  return;  }  i++;  }  }  int lexer(){  lexLen=0;  int flag=1;  getNonBlank();  switch(charClass){  case UNDERLINE:  addChar();  getChar();  while(charClass==LETTER||charClass==DIGIT||charClass==UNDERLINE){  addChar();  getChar();  }  nextToken=Y\_ID;  break;  case LETTER:  addChar();  getChar();  while(charClass==LETTER||charClass==DIGIT||charClass==UNDERLINE){  addChar();  getChar();  }  nextToken=Y\_ID;  checkKeywords(lexeme);  break;  case DIGIT:  addChar();  getChar();  while(charClass==DIGIT){  addChar();  getChar();  }  nextToken=num\_INT;  if(charClass==POINT){  addChar();  getChar();  while(charClass==DIGIT){  addChar();  getChar();  }  nextToken=num\_FLOAT;  }  break;  case UNKNOWN:  flag=checkSymbol(nextChar,next2Char);  getChar();  break;  case EOF:  nextToken=Y\_EOF;  lexeme[0]='E';  lexeme[1]='O';  lexeme[2]='F';  lexeme[3]=0;  break;  }  if(flag!=0){  printf("<%6d, %s\t>\n",nextToken,lexeme);  }  return nextToken;  }  void main(int argc,char\* argv[]){  if(argc<2){  printf("input file is needed.\n");  exit(0);  }  f=fopen(argv[1],"r");  if(f==NULL){  printf("can not open file.\n");  exit(0);  }  getChar();  while(nextToken!=Y\_EOF)  lexer();  } |

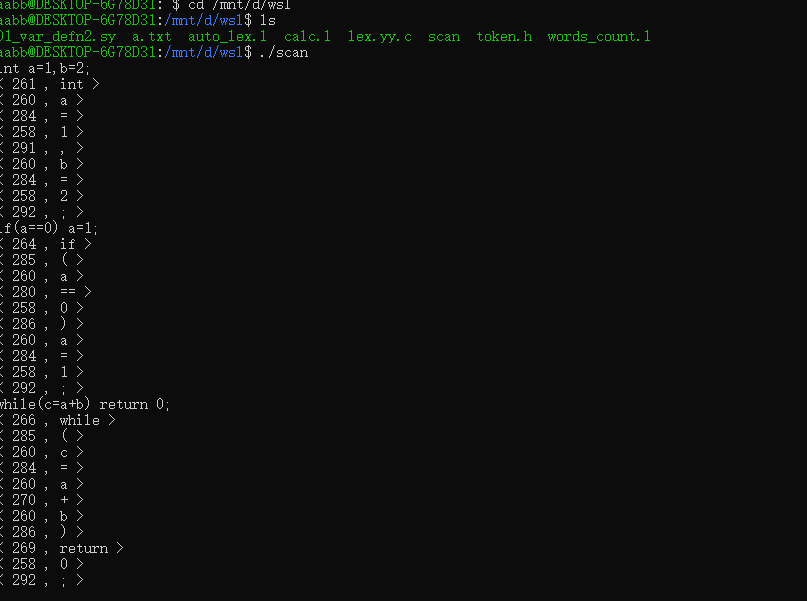
2.以下是利用flex生成的词法分析器：

|  |
| --- |
| %{  enum yytokentype {  num\_INT = 258,  num\_FLOAT = 259,  Y\_ID = 260,  Y\_INT = 261,  Y\_VOID = 262,  Y\_CONST = 263,  Y\_IF = 264,  Y\_ELSE = 265,  Y\_WHILE = 266,  Y\_BREAK = 267,  Y\_CONTINUE = 268,  Y\_RETURN = 269,  Y\_ADD = 270,  Y\_SUB = 271,  Y\_MUL = 272,  Y\_DIV = 273,  Y\_MODULO = 274,  Y\_LESS = 275,  Y\_LESSEQ = 276,  Y\_GREAT = 277,  Y\_GREATEQ = 278,  Y\_NOTEQ = 279,  Y\_EQ = 280,  Y\_NOT = 281,  Y\_AND = 282,  Y\_OR = 283,  Y\_ASSIGN = 284,  Y\_LPAR = 285,  Y\_RPAR = 286,  Y\_LBRACKET = 287,  Y\_RBRACKET = 288,  Y\_LSQUARE = 289,  Y\_RSQUARE = 290,  Y\_COMMA = 291,  Y\_SEMICOLON = 292,  Y\_FLOAT = 293,  Y\_EOF = 294,  Y\_FOR = 295  };  typedef union \_YYLVAL{  int token;  int int\_value;  float float\_value;  char\* id\_name;  }\_YYLVAL;  \_YYLVAL yylval;  %}  DIGIT [0-9]  INT [0-9][0-9]\*  FLOAT {DIGIT}\*(\.{DIGIT}+)?(e|E[+|-]?{DIGIT}+)?  WORD [a-zA-Z]  ID [a-zA-Z\_][a-zA-Z0-9\_]\*  %%  "for" {return Y\_FOR;}  "int" {return Y\_INT;}  "float" {return Y\_FLOAT;}  "void" {return Y\_VOID;}  "const" {return Y\_CONST;}  "if" {return Y\_IF;}  "else" {return Y\_ELSE;}  "while" {return Y\_WHILE;}  "break" {return Y\_BREAK;}  "continue" {return Y\_CONTINUE;}  "return" {return Y\_RETURN;}  {INT} {return num\_INT;}  {FLOAT} {return num\_FLOAT;}  [\_A-Za-z]+[\_a-zA-Z0-9]\* {return Y\_ID;}  "+" { return Y\_ADD; }  "-" { return Y\_SUB; }  "\*" { return Y\_MUL; }  "/" { return Y\_DIV; }  "%" {return Y\_MODULO;}  "<" {return Y\_LESS;}  "<=" {return Y\_LESSEQ;}  ">" {return Y\_GREAT;}  ">=" {return Y\_GREATEQ;}  "!=" {return Y\_NOTEQ;}  "==" {return Y\_EQ;}  "!" {return Y\_NOT;}  "&&" {return Y\_AND;}  "||" {return Y\_OR;}  "=" {return Y\_ASSIGN;}  "(" {return Y\_LPAR;}  ")" {return Y\_RPAR;}  "{" {return Y\_LBRACKET;}  "}" {return Y\_RBRACKET;}  "[" {return Y\_LSQUARE;}  "]" {return Y\_RSQUARE;}  "," {return Y\_COMMA;}  ";" {return Y\_SEMICOLON;}  [ \t\n] {}  "//".\*[\n] {}  %%  int main(int argc, char \*\*argv)  {  int tok;  if(argc<2){  printf("input file is needed.\n");  exit(0);  }  yyin=fopen(argv[1],"r");  if(yyin==NULL){  printf("can not open file.\n");  exit(0);  }  while(tok = yylex())  {  printf("<\t%d\t,\t", tok);  printf("%s\t>\n",yytext);  }  }  int yywrap(){  return 1;  } |

**五、实验运行结果及测试：**

设置将每个所分析的符号输出为类型与值的形式。

输入测试文件，则输出如下所示。



**六、实验结论与总结：**

经过测试，auto\_lex.l最终生成的可执行文件都能够识别常数、运算符、界符、标识符、关键字，并无视单行注释，能够正确进行词法分析。

同样的效果C语言自行编写需要上百行代码，而利用flex协助完成的.l源文件却只需要短短几十行，令人感慨。

**报告评分：**

**指导教师签字：**