C语言程序词法分析（C++实现）实验报告

姓名：熊宇

班级：2018211302

学号：2018210074

指导教师：李文生

目录

**实验题目及要求2**

**实验设备3**

**程序设计说明4**

**实验流程图5**

**实验输入（测试程序）6**

**实验程序7**

**实验运行结果及分析说明16**

**心得体会19**

**LEX源程序20**

1. 实验题目及要求
2. 设计一个C语言词法分析程序；
3. 可以识别出用C语言编写的源程序中的每个单词符号，并以记号的形式输出每个单词符号；
4. 可以识别并跳过源程序中的注释；
5. 可以统计源程序中的语句行数、各类单词个数、以及字符总数，并输出统计结果；
6. 检查源程序中存在的词法错误，并报告错误所在的位置；
7. 对源程序中出现的错误进行适当的恢复，使词法分析可以继续进行，对源程序进行一次扫描，即可检查并报告源程序中存在的所有词法错误。
8. 实验设备

Windows 10 设备机，Visual Stdio Code编程环境，FLEX。

1. 程序设计说明
2. 词法分析程序的作用：①扫描源程序字符流；

②按照源语言的词法规则识别出各类单词符号；

③产生用于语法分析的记号序列；

④词法检查；

⑤创建符号表：将识别出来的标识符插入符号表中；

⑥跳过源程序中的注释和空白，把错误信息和源程序联系起来；

1. 源程序分解单词归类：①关键字；

②变量名；

③数字常量；

④分界符和运算符；

1. 实验思路

根据有限自动机的概念，将整个源代码分解的规则转化为一个一个状态，用switch或if/else将状态之间的转移描述出来即可。这里省略了将常量以及变量名插入到符号表返回指针的那一步。

伪代码描述：

*初始化 状态=0*

*switch(状态):*

*{*

*case 0:*

*接收源码*

*if(符合1规则)*

*处理，状态=1*

*else if(符合2规则)*

*处理，状态=2*

*else*

*...*

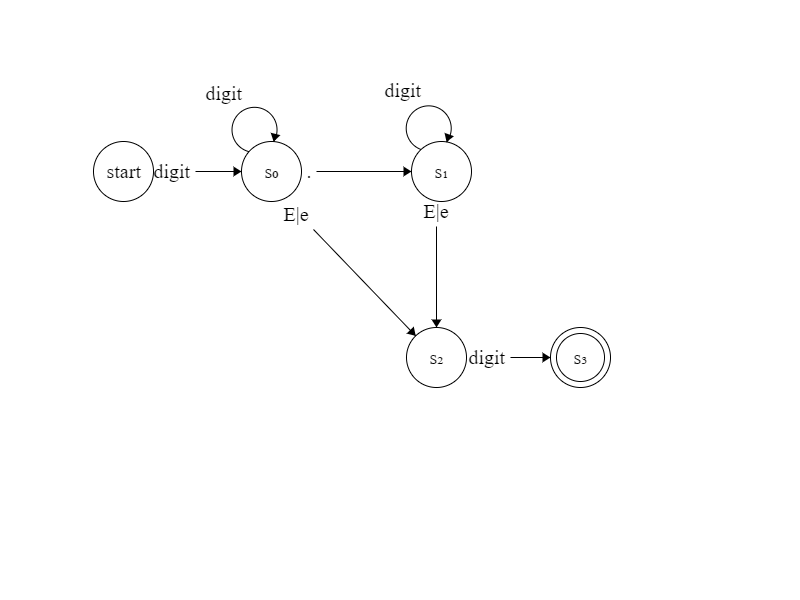
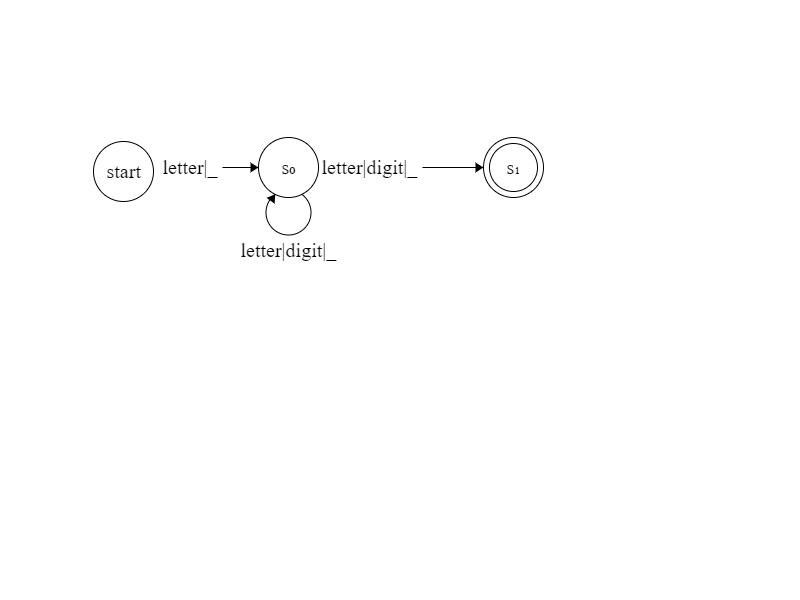
*case 1:*

*case 2:*

*case ...*

*}*

1. 实验流程图



1. 实验输入（测试程序）

*/\**

*Version:1.0*

*User:Septer*

*Last Updated Time:2020/10/06 16:42*

*Todo:Test*

*\*/*

*#define relax 10*

*#include <stdio.h>*

*#include <math.h>*

*int main()*

*{*

*int m,flag,i;*

*scanf("%d", &m);*

*flag = 1;*

*for(i = 2;i <= sqrt(m);i++)*

*{*

*if(m%i == 0)*

*{*

*flag = 0;*

*break;*

*}*

*}*

*if(flag)*

*{*

*//Now have a output.*

*printf("%d is primers.\n", m);*

*}*

*else*

*{*

*//Now have a output.*

*printf("%d is not primers.\n", m);*

*}*

*return 0;*

*}*

1. 实验程序

*#include <iostream>*

*#include <fstream>*

*#include <iomanip>*

*#include <iostream>*

*#include <algorithm>*

*#include <cmath>*

*#include <cstring>*

*#include <vector>*

*#include <map>*

*using namespace std;*

*int NumOfKey = 11; //内置关键字数目*

*char \*KeyWord[] = {*

*"for",*

*"while",*

*"do",*

*"continue",*

*"if",*

*"else",*

*"char",*

*"int",*

*"double",*

*"float",*

*"return"}; //内置关键字数组*

*char \*FName = "F:\\Compile\\WordAnalysis\\test.txt";*

*char Buffer[1026]; //存储一行*

*char Output[200]; //存储输出*

*int NumOfLetter = 0; //字符数目*

*int NumOfKeyWord = 0; //关键字数目*

*int NumOfVarible = 0; //变量名数目*

*int NumOfNumber = 0; //数字常量数目*

*int NumOfOperator = 0; //分界符和操作符数目*

*//位置回退*

*void GoBack(char \*&s)*

*{*

*s--;*

*}*

*//获取下一个字符*

*char GetAnother(char \*&s)*

*{*

*return \*s++;*

*}*

*//判断是否是数字*

*bool IsNum(char s)*

*{*

*if ((s >= '0') && (s <= '9'))*

*return true;*

*else*

*return false;*

*}*

*//判断是否为字母*

*bool IsLetter(char s)*

*{*

*if ((s >= 'a') && (s <= 'z'))*

*return true;*

*else*

*{*

*if ((s >= 'A') && (s <= 'Z'))*

*return true;*

*else*

*return false;*

*}*

*}*

*//判断是否为关键字，如果是返回下标；否则返回0*

*int IsKey(char \*s)*

*{*

*for (int i = 0; i < NumOfKey; i++)*

*{*

*if (strcmp(s, KeyWord[i]) == 0)*

*{*

*return i + 1;*

*break;*

*}*

*}*

*return 0;*

*}*

*//打印结果*

*void PrintRes(int lineNum, char \*in, char \*belong)*

*{*

*cout << lineNum << " <" << in << ", " << belong << ">" << endl;*

*}*

*//一行一行处理*

*void Work(char \*s, int lineNum)*

*{*

*int state = 0; //状态初始化*

*char ch = ' ';*

*int pos = 0; //标记位置*

*while (ch != '\0')*

*{*

*switch (state)*

*{*

*case 0:*

*{*

*ch = GetAnother(s);*

*switch (ch)*

*{*

*//空格不处理*

*case ' ':*

*pos = 0;*

*break;*

*//以下为分解符*

*case '[':*

*case ']':*

*case '(':*

*case ')':*

*case '{':*

*case '}':*

*{*

*char temp[2];*

*temp[0] = ch;*

*PrintRes(lineNum, temp, "-");*

*pos = 0;*

*NumOfOperator++;*

*break;*

*}*

*//除号*

*case '\'':*

*{*

*state = 0;*

*while (((ch = GetAnother(s)) != '\'') && (ch != '\0'))*

*{*

*Output[pos++] = ch;*

*}*

*if (ch == '\0')*

*{*

*//error*

*//PrintRes("Error", Output);*

*}*

*Output[pos] = '\0'; //加\0生成完整字符串*

*PrintRes(lineNum, "string", Output);*

*//输出后重新置pos为0*

*pos = 0;*

*NumOfOperator++;*

*break;*

*}*

*case '"':*

*{*

*state = 0;*

*while (((ch = GetAnother(s)) != '"') && (ch != '\0'))*

*{*

*Output[pos++] = ch;*

*}*

*if (ch == '\0')*

*{*

*//error*

*PrintRes(lineNum, "Error", Output);*

*}*

*else*

*{*

*Output[pos] = '\0';*

*PrintRes(lineNum, "string", Output);*

*pos = 0;*

*NumOfOperator++;*

*}*

*break;*

*}*

*//加号*

*case '+':*

*{*

*state = 0;*

*ch = GetAnother(s);*

*NumOfOperator++;*

*switch (ch)*

*{*

*//++*

*case '+':*

*PrintRes(lineNum, "++", "-");*

*pos = 0;*

*break;*

*//+=*

*case '=':*

*PrintRes(lineNum, "+=", "-");*

*pos = 0;*

*break;*

*default:*

*GoBack(s);*

*PrintRes(lineNum, "+", "-");*

*pos = 0;*

*break;*

*}*

*}*

*//-号*

*case '-':*

*{*

*state = 0;*

*ch = GetAnother(s);*

*NumOfOperator++;*

*switch (ch)*

*{*

*//--*

*case '-':*

*PrintRes(lineNum, "--", "-");*

*pos = 0;*

*break;*

*//-=*

*case '=':*

*PrintRes(lineNum, "-=", "-");*

*pos = 0;*

*break;*

*default:*

*GoBack(s);*

*PrintRes(lineNum, "-", "-");*

*pos = 0;*

*break;*

*}*

*break;*

*}*

*//=号*

*case '=':*

*{*

*state = 0;*

*ch = GetAnother(s);*

*NumOfOperator++;*

*switch (ch)*

*{*

*//==*

*case '=':*

*PrintRes(lineNum, "==", "-");*

*pos = 0;*

*break;*

*default:*

*GoBack(s);*

*PrintRes(lineNum, "=", "-");*

*pos = 0;*

*break;*

*}*

*break;*

*}*

*default:*

*{*

*if (IsNum(ch))*

*{*

*Output[pos++] = ch;*

*NumOfNumber++;*

*state = 2;*

*}*

*else*

*{*

*if (IsLetter(ch) || ch == '\_')*

*{*

*Output[pos++] = ch;*

*NumOfLetter++;*

*state = 1;*

*}*

*}*

*break;*

*}*

*}*

*break;*

*}*

*case 1:*

*{*

*while (true)*

*{*

*ch = GetAnother(s);*

*if (IsLetter(ch) || IsNum(ch) || ch == '\_')*

*{*

*Output[pos++] = ch;*

*}*

*else*

*{*

*Output[pos] = '\0';*

*int flag = IsKey(Output);*

*//判断是否为关键字*

*if (flag == 0)*

*{*

*PrintRes(lineNum, "Variable", Output);*

*NumOfVarible++;*

*}*

*else*

*{*

*PrintRes(lineNum, "KeyWord", Output);*

*NumOfKeyWord++;*

*}*

*//回归初始*

*GoBack(s);*

*pos = 0;*

*state = 0;*

*break;*

*}*

*}*

*break;*

*}*

*case 2:*

*{*

*while (true)*

*{*

*ch = GetAnother(s);*

*if (IsNum(ch))*

*{*

*Output[pos++] = ch;*

*}*

*else*

*{*

*if (ch == '.')*

*{*

*Output[pos++] = ch;*

*state = 3; //进入小数模式*

*break;*

*}*

*else*

*{*

*if (ch == 'E' || ch == 'e')*

*{*

*Output[pos++] = ch;*

*state = 4; //进入科学计数法模式*

*break;*

*}*

*else*

*{*

*Output[pos] = '\0'; //补\0生成完整字符串*

*PrintRes(lineNum, "Number", Output);*

*GoBack(s);*

*pos = 0;*

*state = 0;*

*break;*

*}*

*}*

*}*

*}*

*break;*

*}*

*case 3:*

*{*

*while (true)*

*{*

*ch = GetAnother(s);*

*if (IsNum(ch))*

*{*

*Output[pos++] = ch;*

*}*

*else*

*{*

*if (ch == 'E' || ch == 'e')*

*{*

*Output[pos++] = ch;*

*state = 4; //进入科学计数法模式*

*}*

*else*

*{*

*Output[pos] = '\0'; //补\0生成完整字符串*

*PrintRes(lineNum, "Number", Output);*

*pos = 0;*

*state = 0;*

*break;*

*}*

*}*

*}*

*break;*

*}*

*case 4:*

*{*

*while (true)*

*{*

*ch = GetAnother(s);*

*if (IsNum(ch))*

*{*

*Output[pos++] = ch;*

*}*

*else*

*{*

*Output[pos] = '\0';*

*PrintRes(lineNum, "Number", Output);*

*GoBack(s);*

*pos = 0;*

*state = 0;*

*break;*

*}*

*}*

*break;*

*}*

*default:*

*PrintRes(lineNum, "Error", Output);*

*break;*

*}*

*}*

*}*

*int main()*

*{*

*FILE \*file = fopen(FName, "r");*

*int lineNum = 0;*

*while (NULL != fgets(Buffer, 1024, file))*

*{*

*lineNum++;*

*Work(Buffer, lineNum);*

*}*

*cout << endl;*

*cout << "行数：" << lineNum << endl;*

*cout << "字符数目：" << NumOfLetter << endl;*

*cout << "关键字数目：" << NumOfKeyWord << endl;*

*cout << "变量名数目" << NumOfVarible << endl;*

*cout << "数字常量数目：" << NumOfNumber << endl;*

*cout << "分界符和操作符数目：" << NumOfOperator << endl;*

*system("pause");*

*return 0;*

*}*

1. 实验运行结果及分析说明

*2 <Variable, Version>*

*2 <Number, 1.0>*

*3 <Variable, User>*

*3 <Variable, Septer>*

*4 <Variable, Last>*

*4 <Variable, Updated>*

*4 <Variable, Time>*

*4 <Number, 2020>*

*4 <Number, 10>*

*4 <Number, 06>*

*4 <Number, 16>*

*4 <Number, 42>*

*5 <Variable, Todo>*

*5 <Variable, Test>*

*8 <Variable, define>*

*8 <Variable, relax>*

*8 <Number, 10>*

*10 <Variable, include>*

*10 <Variable, stdio>*

*10 <Variable, h>*

*11 <Variable, include>*

*11 <Variable, math>*

*11 <Variable, h>*

*13 <KeyWord, int>*

*13 <Variable, main>*

*13 <(, ->*

*13 <), ->*

*14 <{, ->*

*15 <KeyWord, int>*

*15 <Variable, m>*

*15 <Variable, flag>*

*15 <Variable, i>*

*16 <Variable, scanf>*

*16 <(, ->*

*16 <string, %d>*

*16 <Variable, m>*

*16 <), ->*

*17 <Variable, flag>*

*17 <=, ->*

*17 <Number, 1>*

*18 <KeyWord, for>*

*18 <(, ->*

*18 <Variable, i>*

*18 <=, ->*

*18 <Number, 2>*

*18 <Variable, i>*

*18 <=, ->*

*18 <Variable, sqrt>*

*18 <(, ->*

*18 <Variable, m>*

*18 <), ->*

*18 <Variable, i>*

*18 <++, ->*

*18 <-, ->*

*18 <), ->*

*19 <{, ->*

*20 <KeyWord, if>*

*20 <(, ->*

*20 <Variable, m>*

*20 <Variable, i>*

*20 <==, ->*

*20 <Number, 0>*

*20 <), ->*

*21 <{, ->*

*22 <Variable, flag>*

*22 <=, ->*

*22 <Number, 0>*

*23 <Variable, break>*

*24 <}, ->*

*25 <}, ->*

*26 <KeyWord, if>*

*26 <(, ->*

*26 <Variable, flag>*

*26 <), ->*

*27 <{, ->*

*28 <Variable, Now>*

*28 <Variable, have>*

*28 <Variable, a>*

*28 <Variable, output>*

*29 <Variable, printf>*

*29 <(, ->*

*29 <string, %d is primers.\n>*

*29 <Variable, m>*

*29 <), ->*

*30 <}, ->*

*31 <KeyWord, else>*

*32 <{, ->*

*33 <Variable, Now>*

*33 <Variable, have>*

*33 <Variable, a>*

*33 <Variable, output>*

*34 <Variable, printf>*

*34 <(, ->*

*34 <string, %d is not primers.\n>*

*34 <Variable, m>*

*34 <), ->*

*35 <}, ->*

*36 <KeyWord, return>*

*36 <Number, 0>*

*37 <}, ->*

*行数：37*

*字符数目：52*

*关键字数目：7*

*变量名数目45*

*数字常量数目：12*

*分界符和操作符数目：36*

*Press any key to continue . . .*

分析说明：该C语言词法分析程序

①可以识别出用C语言编写的源程序中的每个单词符号，并以记号的形式输出每个单词符号；

②可以识别并跳过源程序中的注释；

③可以统计源程序中的语句行数、各类单词个数、以及字符总数，并输出统计结果；

④检查源程序中存在的词法错误，并报告错误所在的位置；

⑤对源程序中出现的错误进行适当的恢复，使词法分析可以继续进行，对源程序进行一次扫描，即可检查并报告源程序中存在的所有词法错误。

1. 心得体会

经过这次词法分析程序的编写，我认识到了词法分析的重要性。他独立的一遍来作为语法分析的子程序，又与语法分析作为协同程序。所谓协同程序就是说他与语法分析程序在同一遍中，以生产者和消费者的关系同步运行。

在程序的编译运行过程中，词法分析的存在避免了中间文件，省去了取送符号的工作，有利于提高编译程序的效率。这里值得一提的是符号表，符号表并非任何语言都有。对于块结构化语言就没有，只有非块结构化语言才有。因为块结构化语言的变量有作用域，需要声明位置；变量与变量之间类型不同但可以同名，对于符号表的作用上来说，没有办法区分作用域。因此块结构化语言没有符号表，本实验所编写的C语言词法分析就没有符号表。

另外，本实验将词法分析独立分离出来拥有诸多好处。如可以简化设计，容易识别并去除空格、注释，使语法分析程序致力于语法分析，结构清晰，易于实现；可以改进编译程序的效率，利用专门的读字符和处理记号的技术构造更有效的词法分析程序；还可以加强编译程序的可移植性，在词法分析程序中处理特殊的或非标准的符号即可等。在本次实验过程中，对这些好处有了更加实际的体会。

此外，本次实验我接触了利用LEX自动生成词法分析程序，在LEX自动生成和自己手动实现的对比下，我体会到了前辈们的智慧和动手能力，这将激励着我不断进取、保持谦虚地学习。LEX源程序在报告结尾附。

在本次实验过程中，我对自动机、词法分析的相关知识有了更深一步的理解和运用，同时提高了自己的编程能力、统筹能力和全局观念，使我受益匪浅。

1. LEX源程序

*%{*

*#include<math.h>*

*#include<stdlib.h>*

*#include<stdio.h>*

*%}*

*DIGIT [0-9]*

*ID [a-z][a-z0-9]\**

*%%*

*{DIGIT}+ {printf("整数: %s(%d)\n",yytext,atoi(yytext));}*

*{DIGIT}+"."{DIGIT}+ {printf("实数: %s(%g)\n",yytext,atof(yytext));}*

*if|then|begin|end|program|while|repeat {printf("关键字: %s\n",yytext);}*

*{ID} {printf("标识符: %s\n",yytext);}*

*"+"|"-"|"\*"|"/" {printf("运算符: %s\n",yytext);}*

*"{"[^}\n]\*"}";*

*[\t\n\x20]+;*

*. {printf("不能识别的字符:%s\n",yytext);}*

*%%*

*int main(int argc,char \*\*argv)*

*{*

*++argv;*

*--argc;*

*if(argc>0) yyin=fopen(argv[0],"r");*

*else yyin=stdin;*

*yylex();*

*return 0;*

*}*

*int yywrap()*

*{*

*return 1;*

*}*