

课 程 设 计 说 明 书

设计题目： 编译原理课程设计

专 业： 计算机科学与技术 班级： 2013-4

设计人： 林宇强

学 号： 201301050415

山 东 科 技 大 学

2016 年 6 月 24 日

课程设计任务书

学院 信息科学与工程学院 专业 计算机科学与技术 班级 2013-4 姓名 林宇强

一、课程设计题目： 编译原理课程设计

二、课程设计主要参考资料

- (1) 韩太鲁等, 编译原理. 石油大学出版社.2007.9
- (2)
- (3)

三、课程设计应解决的主要问题:

- (1) 词法分析之基于 Lex 实现词法分析
- (2) 词法分析之基于文法的实现
- (3) 语法分析之 LL1 分析法实现
- (4)

四、课程设计相关附件（如：图纸、软件等）:

- (1) vc++6.0
- (2) Parser Generator
- (3) Code: Blocks13.12

五、任务发出日期: 2016-5-7 课程设计完成日期: 2016-6-24

指导教师签字: 系主任签字:

指导教师对课程设计的评语

成绩：_____

指导教师签字：_____

年 月 日

目录

词法分析之基于 Lex 实现词法分析.....	3
一、设计目的.....	3
二、设计要求.....	3
三、设计说明.....	3
1. 需求分析.....	3
2. 概要设计.....	4
3. 详细设计.....	6
四、运行结果及分析.....	7
1. 测试数据.....	7
2. 测试输出的结果.....	8
3. 设计与思考.....	9
五、总结.....	9
词法分析之基于文法的实现.....	10
一、设计目的.....	10
二、设计要求.....	10
三、设计说明.....	10
1. 需求分析：.....	10
2. 概要设计.....	11
3. 详细设计.....	13
四、运行结果及分析.....	16
1.测试数据.....	16
2.测试输出的结果.....	17
3.设计和思考.....	18
五、总结.....	18

语法分析之 LL1 分析法实现.....	19
一、设计目的	19
二、设计要求	19
三、设计说明	21
1. 需求分析:	21
2. 概要设计	21
3. 详细设计	22
四、运行结果及分析	25
1.测试数据	25
2.测试输出的结果	26
3.设计和思考	27
五、总结	28

词法分析之基于 Lex 实现词法分析

一、设计目的

通过编写并上机调试一个词法分析程序，掌握在对程序设计语言的源程序进行扫描的过程中，将其分解成各类单词的词法分析方法。

二、设计要求

编制一个读单词过程，从输入的源程序中，识别出各个具有独立意义的单词，即基本保留字、标识符、常数、运算符、分隔符五大类。并依次输出各个单词的内部编码及单词符号自身值。

（遇到错误时可显示“Error”，然后跳过错误部分继续显示）

三、设计说明

1. 需求分析

a) 输入及其范围

Lex 输入文件由 3 个部分组成：定义集（definition），规则集（rule）和辅助程序集（auxiliary routine）或用户程序集（user routine）。这三个部分由位于新一行第一列的双百分号分开，因此，Lex 输入文件的格式如下

```
{definitions}
```

```
%%
```

```
{rules}
```

```
%%
```

```
{auxiliary routines}
```

范围：

识别保留字：const, var, begin, end, read, while, call, writeln 等

保留字类别码为 1。

运算符包括：+、-、*、/、=、>、<、>=、<=、!= ；类别码为 2。

界符包括：,、;、{、}、(、)； 类别码为 3。

常数为无符号整形数；单词类别码为 4。

其他的都识别为标识符；单词类别码为 5。

错误字符 类别码为 6。

b) 输出形式

([数字], “单词”) 数字代表所识别的单词所属的类型。

c) 程序功能

读取文件中的源程序，然后对源程序分析，输出分析结果。

d) 测试数据

将测试数据写在文本文件中，测试为程序的输入数据。

2. 概要设计

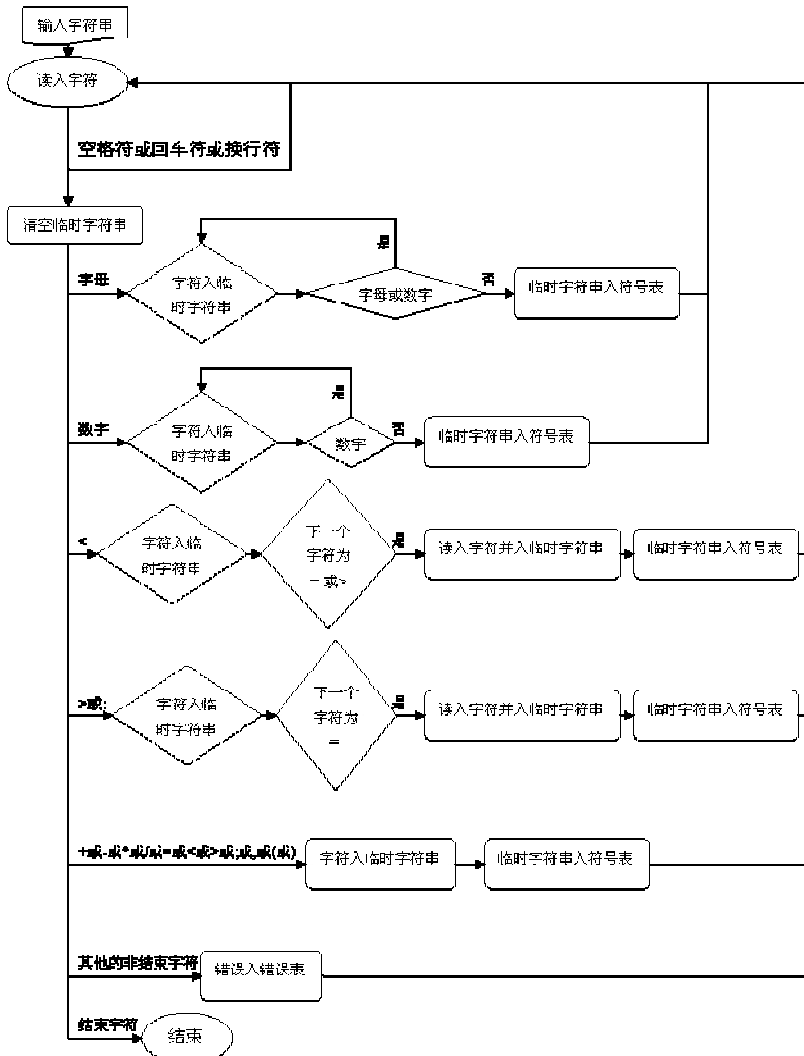
a) 数据类型的定义

```
%{
    #include <stdio.h>
    #include <stdlib.h>
    int count = 0;
}%

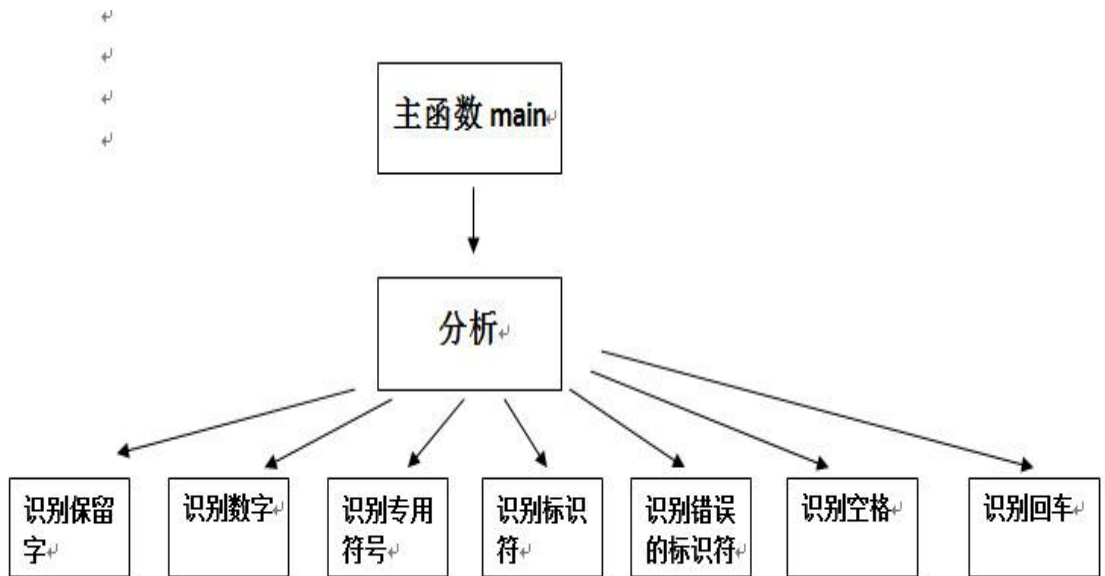
delim [" "\n\t]
whitespace {delim}+
operator \+|\-|\*|\/|:=|#|=
//operator \+|\-|\*|\/|:=|>|=|<|=|#|=
reservedWord
[cC][oO][nN][sS][tT][vV][aA][rR][pP][rR][oO][cC][eE][dD][uU][rR]
[eE][bB][eE][gG][iI][nN][eE][nN][dD][iI][fF][tT][hH][eE][nN][wW]
[hH][iI][lL][eE][dD][oO][rR][eE][aA][dD][cC][aA][lL][lL][wW][rR]
```

][iI][tT][eE][wW][rR][iI][tT][eE][lL][nN]
 delimiter [, \, ;]
 constant ([0-9])+
 identifier [A-Za-z]([A-Za-z][0-9])*
 %%

b) 主程序流程



c) 模块间的调用关系



3. 详细设计

主体代码部分：

```

%%
{reservedWord} {count++;printf("%d\t(1, '%s')\n",count,yytext);}
{operator} {count++;printf("%d\t(2, '%s')\n",count,yytext);}
{delimiter} {count++;printf("%d\t(3, '%s')\n",count,yytext);}
{constant} {count++;printf("%d\t(4, '%s')\n",count,yytext);}
{identfier} {count++;printf("%d\t(5, '%s')\n",count,yytext);}
{whitespace} { /* do nothing*/ }
    
```

%%

```
void main()
```

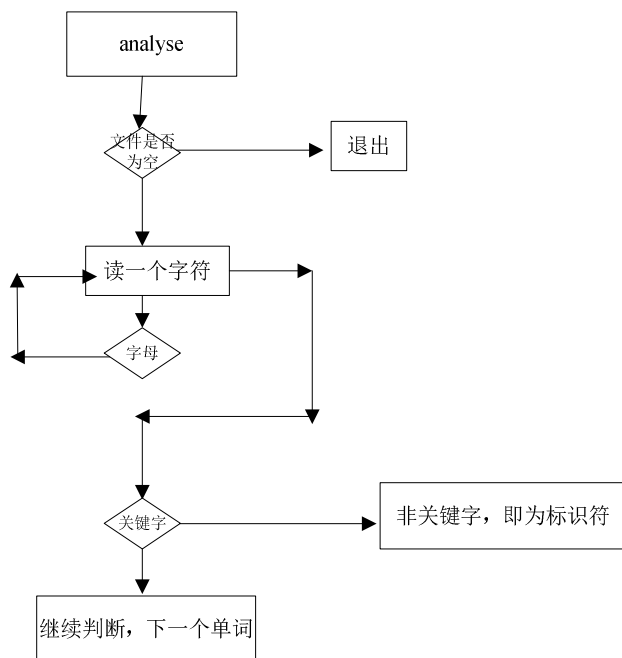
```

{
    printf("词法分析器输出类型说明:\n");
    printf("1: 保留字\n");
    printf("2: 运算符\n");
    printf("3: 分界符\n");
    printf("4: 常 数\n");
    printf("5: 标识符\n");
    printf("\n");
    
```

```

yyin=fopen("example.txt","r");
    yylex(); /* start the analysis*/
fclose(yyin);
system("PAUSE");/*暂停停， 使 DOS 窗口停住*/
}
int yywrap()
{
    return 1;
}

```



四、运行结果及分析

1. 测试数据

在 example.txt 文件中写入如下数据。

```

const a=10;
var b,c;

```

```

procedure p;
begin
    c:=b+a;
end;

begin
    read(b);
    while b#0 do
    begin
        call p;
        writeln(2*c);
        read(b);
    end
end.

```

2. 测试输出的结果



3. 设计与思考

基于 lex 的词法分析，一个模块定义好了之后，其他模块也就出来了。难点在于正则表达式的设计，每个模块都有定义集，规则集和辅助程序集。而且第一部分用 “%{” 和 “}%” 括起来。第一和第三个部分为 C 语言的代码和函数定义，第二个部分为一些规则。

五、总结

通过本次实验，学会了基于 Lex 的词法分析器构造方法。在实验过程中，万事开头难，刚开始不知道怎么做，以及不知道如何使用老师给的软件，在仔细分析了需求之后，查找了一些关于 Parser Genarator 2 软件的使用方法，但是在使用过程中错误地使用 .y 后缀名导致 Lex 编译总是出错，后来看教程带着尝试的心理改成了 .l 后缀名才终于成功让 Lex 程序跑起来，正确生成编译器 C 代码。

有了编译器 C 代码之后打算开始用 Code::Block 编译，但是发现其中的 yyleh.h 类库找不到，后来用 VS2015 也一直出现问题，是在没办法，只好按照老师给的 VC++ 的教程安装 VC++6.0 并导入 yylex.h 类库，最终获得了成功，很是欣喜。

词法分析之基于文法的实现

一、设计目的

通过设计编制调试一个具体的词法分析程序，加深对词法分析原理的理解。并掌握在对程序设计语言源程序进行扫描过程中将其分解为各类单词的词法分析方法。

二、设计要求

编制一个读单词过程，从输入的源程序中，识别出各个具有独立意义的单词，即基本保留字、标识符、常数、运算符、分隔符五大类。并依次输出各个单词的内部编码及单词符号自身值。

（遇到错误时可显示“Error”，然后跳过错误部分继续显示）

三、设计说明

1. 需求分析：

a) 输入及其范围

识别保留字：IF、THEN、ELSE、GOTO 等，保留字类别码为 K。

其他的都识别为标识符；单词类别码为 I。

常数为无符号整形数；单词类别码为 C。

运算符包括：+、-、*、/、=、>、<、>=、<=、!=；类别码为 O。

界符符包括：,、;、{、}、(、)；类别码为 P。

结束标号 L

b) 输出形式

预处理文件和二元式表 txt 输出文件和控制台输出。

c) 程序功能

词法分析器的功能是输入源程序，输出单词符号二元式。

d) 测试数据

测试输入的程序为：

```
b=100;
101:a=2*(1+3);
IF(b>10)
    THEN a=1;
ELSE IF(b>=5)
    THEN a=2;
ELSE GOTO 101#
```

2. 概要设计

a) 数据类型的定义

```
FILE    *fp;

FILE    *out, *in;

charch;

charfilename[50];

char*keyword[4] = { "IF", "THEN", "ELSE", "GOTO" };

char*operatornum[4] = { "+", "-", "*", "/" };

char*comparison[6] = { ">", "<", ">=", "<=", "=", "<>" };

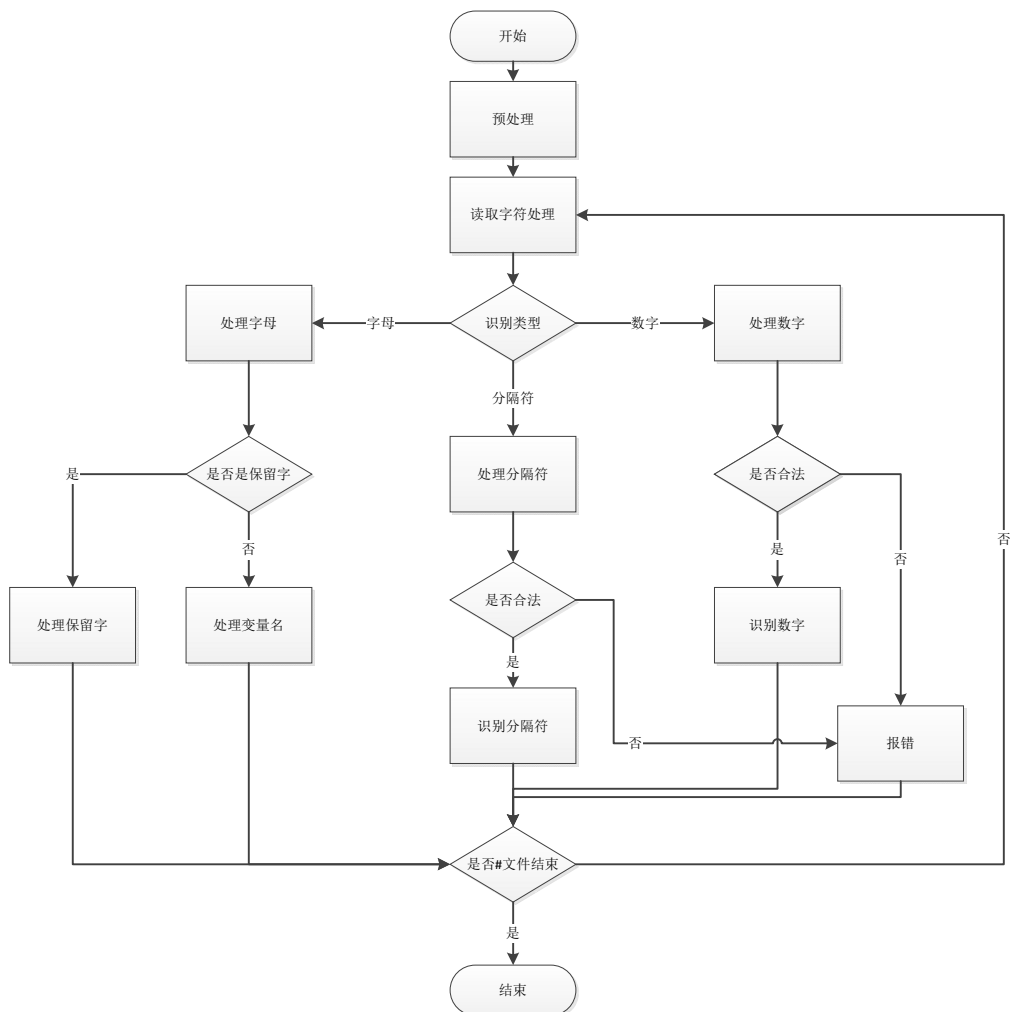
char*interpunction[5] = { ",", ":", "(", ")", ";" };

int  x = -1, y = -1, z = -1;
```

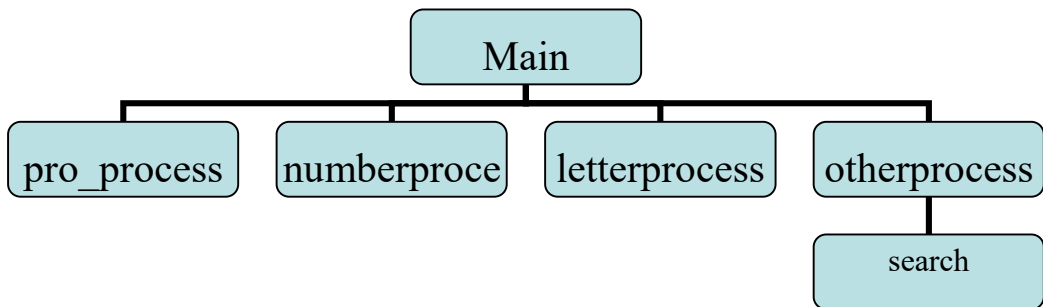
```
charK[Klen][10], P[Plen][2], I[Ilen][20], O[Olen][2];
```

```
int C[Clen], L[Llen];
```

b) 主程序流程



c) 模块间调用关系



3. 详细设计

//处理字母，可能是标识符或者变量

```

char letterprocess( char ch )
{
    int i = -1, j = -1, k, a = 1;
    char letter[10], temp[10];
    while ( isalnum( ch ) != 0 )
    {
        letter[++i] = ch;
        ch = fgetc( fp );
    }
    letter[i + 1] = '\0';
    if ( search( letter, 1 ) )
    {
        printf( "(K,%5s)\n", letter );
        fprintf( out, "(K,%s)\n", letter );
        if ( strcmp( letter, "GOTO" ) == 0 )
        {
            ch = fgetc( fp );
            while ( isdigit( ch ) != 0 )
            {
                temp[++j] = ch;
                ch = fgetc( fp );
            }
            temp[j + 1] = '\0';
        }
    }
}
    
```



```

        printf( "(L,%5s)\n", temp );
        fprintf( out, "(L,%s)\n", temp );
        L[++y] = atoi( temp );
    }
}
else
{
    printf( "(I,%5s)\n", letter );
    fprintf( out, "(I,%s)\n", letter );
    for ( k = 0; k < 10; k++ )
    {
        if ( strcmp( letter, I[k] ) == 0 )
            a = 0;
    }
    if ( a == 1 )
        strcpy( I[++x], letter );
}
return(ch);
}
// 处理数字
char numberprocess( char ch )
{
    int i = -1, n, temp, a = 1;
    char num[20];
    while ( isdigit( ch ) != 0 )
    {
        num[++i] = ch;
        ch = fgetc( fp );
    }
    if ( isalpha( ch ) != 0 )
    {
        while ( isalpha( ch ) != 0 )
        {
            num[++i] = ch;
            ch = fgetc( fp );
        }
        num[i + 1] = '\0';
        printf( "非法标识符: %s\n", num );
    }
}

```

```

        fprintf( out, "非法标识符: %s\n", num );
    }
    if ( ch == ':' )
    {
        num[i + 1] = '\0';
        printf( "(L,%5s)\n", num );
        fprintf( out, "(L,%s)\n", num );
        L[++y] = atoi( num );
    }
    else
    {
        num[i + 1] = '\0';
        printf( "(C,%5s)\n", num );
        fprintf( out, "(C,%s)\n", num );
        temp = atoi( num );
        for ( n = 0; n < 10; n++ )
        {
            if ( temp == C[n] )
                a = 0;
        }
        if ( a == 1 )
            C[++z] = temp;
    }
    return(ch);
}
// 处理其他情况
char otherprocess( char ch )
{
    int i = -1;
    char other[10];
    if ( ch == '(' )
    {
        other[++i] = ch;
        ch = fgetc( fp );
    }
    while ( isalnum( ch ) == 0 && ch != ' ' && ch != '(' && ch !=
    ')') )
    {

```

```

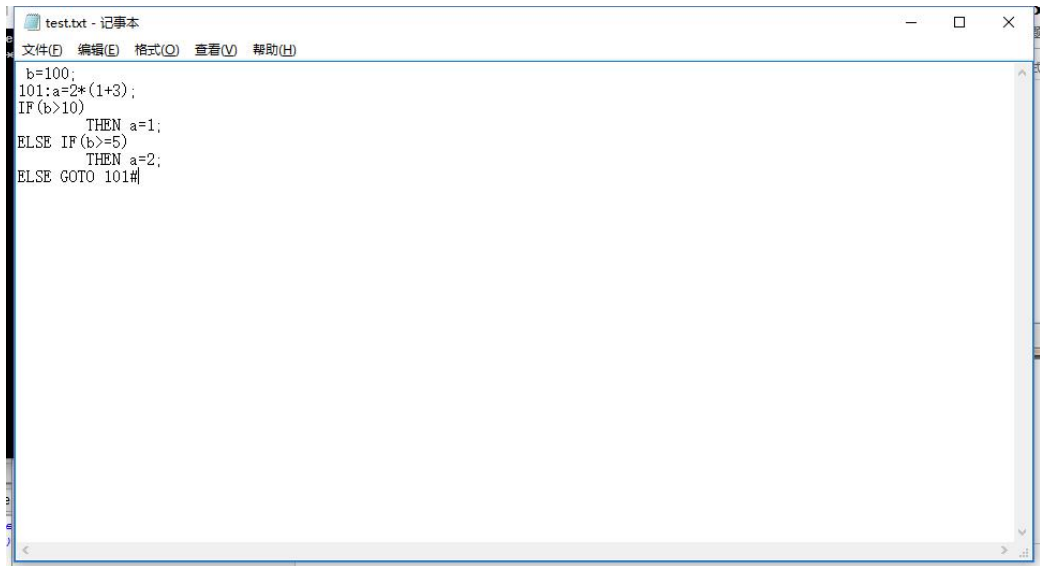
        other[++i] = ch;
        ch        = fgetc( fp );
    }
    if ( ch == ' ' )
    {
        other[++i] = ch;
        ch        = fgetc( fp );
    }
    other[i + 1] = '\0';
    if ( search( other, 4 ) )
    {
        printf( "(P,%5s)\n", other );
        fprintf( out, "(P,%s)\n", other );
    }
    else if ( search( other, 2 ) || search( other, 3 ) )
    {
        printf( "(0,%5s)\n", other );
        fprintf( out, "(0,%s)\n", other );
    }
    else
    {
        printf( "非法字符: %s\n", other );
        fprintf( out, "非法字符: %s\n", other );
    }
    return(ch);
}

```

四、运行结果及分析

1.测试数据

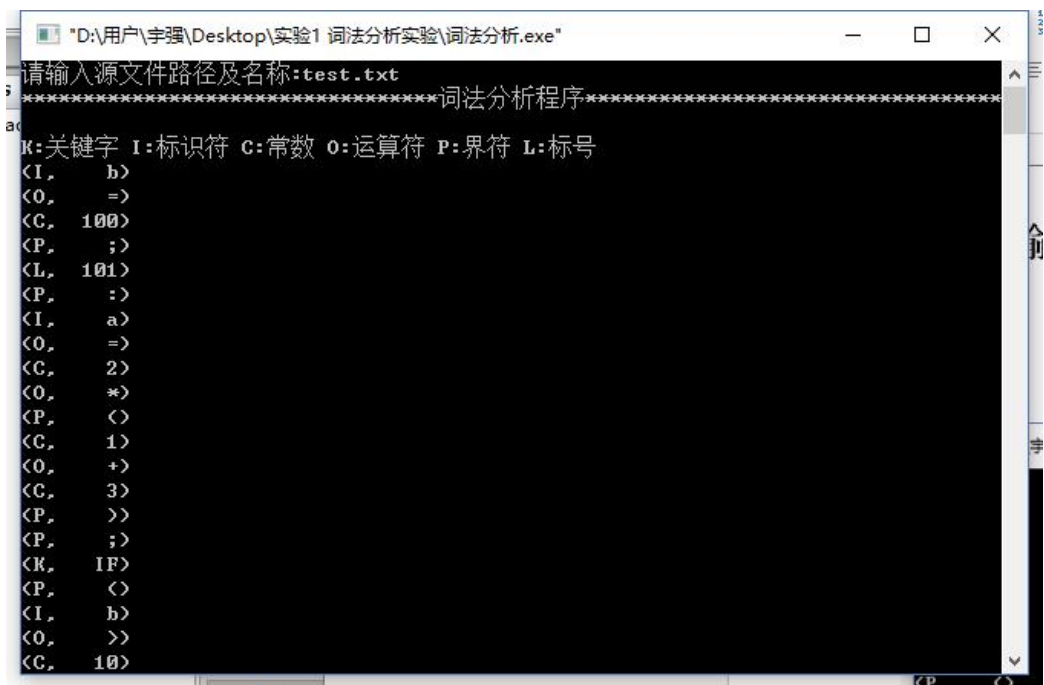
测试数据为 test.txt 文件内的数据：



```

b=100;
101:a=2*(1+3);
IF (b>10)
    THEN a=1;
ELSE IF (b>=5)
    THEN a=2;
ELSE GOTO 101#
    
```

2.测试输出的结果



```

"D:\用户\字强\Desktop\实验1 词法分析实验\词法分析.exe"
请输入源文件路径及名称:test.txt
*****词法分析程序*****
K:关键字 I:标识符 C:常数 O:运算符 P:界符 L:标号
(I, b)
(O, =)
(C, 100)
(P, ;)
(L, 101)
(P, :)
(I, a)
(O, =)
(C, 2)
(O, *)
(P, (<)
(C, 1)
(O, +)
(C, 3)
(P, > >)
(P, ;)
(K, IF)
(P, (<)
(I, b)
(O, >)
(C, 10)
    
```

```

D:\用户\宇强\Desktop\实验1 词法分析实验\词法分析.exe
<C, 10>
<P, >>
<K, THEN>
<I, a>
<O, =>
<C, 1>
<P, ;>
<K, ELSE>
<K, IF>
<P, <>
<I, b>
<O, >=>
<C, 5>
<P, >>
<K, THEN>
<I, a>
<O, =>
<C, 2>
<P, ;>
<K, ELSE>
<K, GOTO>
<L, 101>
输出结果保存在二元式表.txt文件中,请打开查看。
词法分析结束!
请按任意键继续. . .
  
```

3.设计和思考

本程序的设计是基于 C 实现的词法分析程序，对整个程序分模块进行开发，并对源程序预处理和最终结果分别在控制台和输出文件中输出，整个程序的思路就是依据编译原理的理论知识，但是实际使用技术都是 C 最基本的字符串解析和处理方法，很久没用 C 编程，也在这次实验中好好复习了 C 的语法。

五、总结

经过努力，终于完成了本次实验，在实验中遇到了诸多问题，经常发生空指针，数组越界等很基础的错误，不过经过不断地调试，逐渐把整个程序运行起来，也通过这次实验，对书本的理论知识，词法分析流程和方法有了较为深入的了解。

语法分析之 LL1 分析法实现

一、设计目的

根据某一文法编制调试 LL（1）分析程序，以便对任意输入的符号串进行分析。本次实验的目的主要是加深对预测分析 LL（1）分析法的理解。

二、设计要求

程序输入/输出示例：

对下列文法，用 LL（1）分析法对任意输入的符号串进行分析：

原文法：

$$E \rightarrow E+T \mid E-T \mid T$$

$$T \rightarrow T * F \mid T / F \mid F$$

$$F \rightarrow id \mid (E) \mid num$$

其中： id: a-f, A-F, num:0-9

消左递归：

$$E \rightarrow TA \quad A \rightarrow +TA \quad A \rightarrow -TA \quad A \rightarrow e$$

$$T \rightarrow FB \quad B \rightarrow *FB \quad B \rightarrow /FB \quad B \rightarrow e$$

$$F \rightarrow i \quad F \rightarrow (E) \quad F \rightarrow n$$

其中： i:id, n:num, e:epsilon $E \rightarrow TG$

FIRST 集和 FOLLOW 集：

	TA	+TA	-TA	e	FB	*FB	/FB	e	i	(E)	n
FIRST	i, (, n	+	-	e	i, (, n	*	/	e	i	(n

	E	A	T	B	F
FOLLOW	\$,)	\$,)	+, -, \$,)	+, -, \$,)	*, /, +, -, \$,)

输出的格式如下：

(1) 输入一以#结束的符号串(包括+—*/ () i#):

(2) 输出过程如下：

栈	输入	输出
\$E	(a-1)*(3+4/2)+((8*2))\$	E->TA

(3) 输入符号串为非法符号串(或者为合法符号串)

注意：

1. 表达式中允许使用运算符(+—*/)、分割符(括号)、字符 i，结束符#；
2. 如果遇到错误的表达式，应输出错误提示信息（该信息越详细越好）；
3. 测试用的表达式可以事先放在文本文件中，一行存放一个表达式，同时以分号分割。同时将预期的输出结果写在另一个文本文件中，以便和输出进行对照；

三、设计说明

1. 需求分析：

a) 输入及其范围

输入为文法，表达式中允许使用运算符（+*/）、分割符（括号）、字符 a。

b) 输出形式

栈	输入	输出
\$E	(a-1)*(3+4/2)+((8*2))\$	E→TA

c) 程序功能

根据输入的文法进行分析，利用 LL(1) 控制程序根据显示栈栈顶内容、向前看符号以及 LL(1) 分析表，对输入符号串自上而下的分析过程

d) 测试数据

输入：文件“fin.txt”输入待分析串

输出：命令行界面输出预测分析表，LL(1)分析过程输出至“fout.txt”

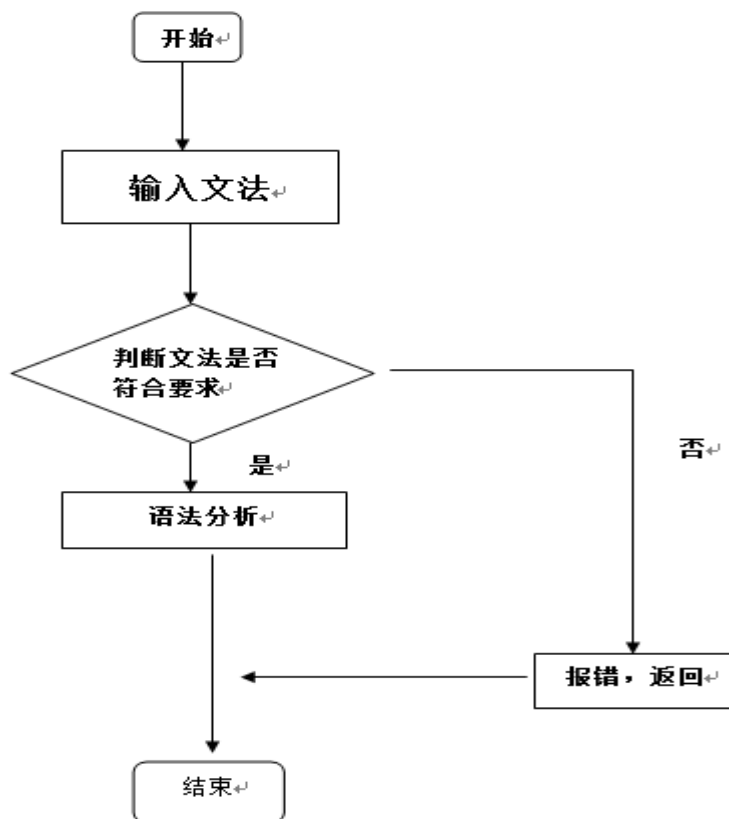
2. 概要设计

a) 数据类型的定义

vector<vector<string>> table(5,vector<string>(9)) //预测分析表


```
vector<string> G           //消除左递归后的文法产生式
map<char, int> index      //文法符号到下标的转换字典
string terminal("in+*/()$") //终结符
string nonTerminal("EATBF") //非终结符
vector<string> First      // 产生式右部符号串的 first 集
vector<string> Follow     //非终结符的 follow 集
```

b) 主程序流程



3. 详细设计

```
1. int main()
{
```

```

for(文法 G 每个产生式 itG, itFirst 为其右部符号串的 first 集) {

    x = itG 左部非终结符号的下标;

    for(itFirst 中的每个终结符号 first) {

        y = 终结符号 first 的下标;

        把 itG 加入分析表表 G[x][y];

    }

if(终结符号 first == epsilon)

    for(Follow 集中的每个符号 follow) {

        y = follow 的下标;

        把 itG 加入分析表 G[x][y];

    }

    }

    for (所有非终结符号的 Follow 集)

if (对应表项为空)

    写入 synch;

    将分析表输出到命令行界面;

return analysis();

}

2. int analysis(void)

```

```
{
    从文件 fin.txt 读取待分析串到 s;

    s 末尾加 '$';

    分析栈 vector<char> analyStack;

    向栈压入 '$'、'E';

    ip 指向 s 的第一个字符;

    do{

        top 是栈顶符号;

        cur 是 ip 所指向的输入符号;

        if (cur 是字母)  cur = 'i';

        if (cur 是数字)  cur = 'n';

        if (top 是终结符号或 '$') {

            if (top == cur) {从栈顶弹出 cur; ip 前移一个位置; }

            else error;

        }

        else{

            x = top 对应下标;      y = cur 对应下标;

            产生式 production = table[x][y];

            if (production 非空) {
```

```
        栈顶弹出 cur;

        把 production 右部逆序压栈;

        输出 production;

    }

    else error;

}

while (top != '$');

}
```

四、运行结果及分析

1.测试数据

fin. txt 文件入字符串: $(a-1)*(3+4/2)+((8*2))$

2.测试输出的结果

```

D:\用户\宇强\Desktop\语法分析\LL(1)\LL(1).exe
预测分析表:
      i      n      +      -      *      /      <      >      $
E      E->TA    E->TA                                E->TA    synch    synch
A                                A->+TA    A->-TA                                A->e    A->e
T      T->FB    T->FB                                T->FB    synch    synch
B      B->e    B->e    B->*FB    B->/FB    B->e    B->e
F      F->i    F->n    synch    synch    synch    synch    F-><E>    synch    synch

成功读取待分析串:
<a-1>*<3+4/2>+<(8*2)>

分析结果已输出至 fout.txt.

Process returned 0 (0x0)   execution time : 0.770 s
Press any key to continue.
    
```

输出文件:

```

fout.txt - 记事本
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)

栈      输入      输出
$E      (a-1)*<3+4/2>+<(8*2)>$    E->TA
$AT      (a-1)*<3+4/2>+<(8*2)>$    T->FB
$ABF      (a-1)*<3+4/2>+<(8*2)>$    F->(E)
$AB)E(    (a-1)*<3+4/2>+<(8*2)>$
$AB)E      a-1)*<3+4/2>+<(8*2)>$    E->TA
$AB)AT      a-1)*<3+4/2>+<(8*2)>$    T->FB
$AB)ABF      a-1)*<3+4/2>+<(8*2)>$    F->i
$AB)ABi      a-1)*<3+4/2>+<(8*2)>$
$AB)AB      -1)*<3+4/2>+<(8*2)>$    B->e
$AB)A      -1)*<3+4/2>+<(8*2)>$    A->-TA
$AB)AT-      -1)*<3+4/2>+<(8*2)>$
$AB)AT      1)*<3+4/2>+<(8*2)>$    T->FB
$AB)ABF      1)*<3+4/2>+<(8*2)>$    F->n
$AB)ABn      1)*<3+4/2>+<(8*2)>$
$AB)AB      )*<3+4/2>+<(8*2)>$    B->e
$AB)A      )*<3+4/2>+<(8*2)>$    A->e
$AB)      )*<3+4/2>+<(8*2)>$
$AB      *<3+4/2>+<(8*2)>$    B->*FB
$ABF*      *<3+4/2>+<(8*2)>$
$ABF      (3+4/2)+<(8*2)>$    F->(E)
$AB)E(    (3+4/2)+<(8*2)>$
$AB)E      3+4/2)+<(8*2)>$    E->TA
$AB)AT      3+4/2)+<(8*2)>$    T->FB
$AB)ABF      3+4/2)+<(8*2)>$    F->n
$AB)ABn      3+4/2)+<(8*2)>$
$AB)AB      +4/2)+<(8*2)>$    B->e
    
```

```
fout.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

$AB)ABn      3+4/2)+(8*2))$
$AB)AB        +4/2)+(8*2))$      B->e
$AB)A         +4/2)+(8*2))$      A->+TA
$AB)AT+       +4/2)+(8*2))$
$AB)AT        4/2)+(8*2))$      T->FB
$AB)ABF       4/2)+(8*2))$      F->n
$AB)ABn       4/2)+(8*2))$
$AB)AB        /2)+(8*2))$      B->/FB
$AB)ABF/      /2)+(8*2))$      F->n
$AB)ABF       2)+(8*2))$
$AB)ABn       2)+(8*2))$
$AB)AB        )+(8*2))$      B->e
$AB)A         )+(8*2))$      A->e
$AB)          )+(8*2))$
$AB)          +((8*2))$      B->e
$A            +((8*2))$      A->+TA
$AT+          +((8*2))$
$AT           ((8*2))$      T->FB
$ABF          ((8*2))$      F->(E)
$AB)E(        ((8*2))$
$AB)E         (8*2))$      E->TA
$AB)AT        (8*2))$      T->FB
$AB)ABF       (8*2))$      F->(E)
$AB)AB)E(     (8*2))$
$AB)AB)E      8*2))$      E->TA
$AB)AB)AT     8*2))$      T->FB
$AB)AB)ABF    8*2))$      F->n
```

```
fout.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

$A            +((8*2))$      A->+TA
$AT+          +((8*2))$
$AT           ((8*2))$      T->FB
$ABF          ((8*2))$      F->(E)
$AB)E(        ((8*2))$
$AB)E         (8*2))$      E->TA
$AB)AT        (8*2))$      T->FB
$AB)ABF       (8*2))$      F->(E)
$AB)AB)E(     (8*2))$
$AB)AB)E      8*2))$      E->TA
$AB)AB)AT     8*2))$      T->FB
$AB)AB)ABF    8*2))$      F->n
$AB)AB)ABn    8*2))$
$AB)AB)AB     *2))$      B->*FB
$AB)AB)ABF*   *2))$      F->n
$AB)AB)ABF    2))$
$AB)AB)ABn    2))$
$AB)AB)AB     ))$      B->e
$AB)AB)A      ))$      A->e
$AB)AB)       ))$
$AB)AB        )$      B->e
$AB)A         )$      A->e
$AB)          )$
$AB)          )$      B->e
$A            )$      A->e
$              )$
$              )$
```

3.设计和思考

主要的难点在于对 LL (1) 的理解部分，消除二义性、消除左递归、提取左因子，判断是否为 LL (1) 文法，然后开始整理思路进行编码阶段。开始要对错误的文法进行分析，并提示详细的错误信息。思考之后实现了表达式中允许使用运算符 (+-* /)、分割符 (括号)、字符 a。

五、总结

本次课程设计是本周实验来难点最大的一次作业,首先需要温习 LL(1) 的知识,如何消除左递归,区别二义性文法,以及对文法的分析。在实验的过程中,最重要的还是要理顺思路,想好解决办法,这也是我经过不断实验总结出的自我思考的方法。然后就进入了编码阶段,此次编码也有一定的难度,在代码量以及代码的整体设计上都有了提升,也是最值得思考的地方。最后,通过实验报告的书写,以及参考资料的查找,对今后的学习和工作都有很大的帮助。