# 实验 2：语法分析

## 1.实验目的

编写一个 LL（1）分析程序，实现对词法分析程序所提供的单词序列的语法检查和结构分析。

## 2.实验要求

利用 LL（1）分析法分析程序，并对简单语言进行语法分析。

### 2.1 待分析的语言语法

首先，根据 c 语言文法消除左递归，简化优化为 LL(1)文法后设计自定义文法产生式如下:(当前文法以英文为准，汉字为 c 语言文法)

1,A->B <源程序>→<函数体>//zwx：为了降低难度，暂时不考虑外部声明和程序外声明情况

2,B->CD(E)J <函数体>→<数据类型><标识符>（<形参>）<复合语句>

3,C->kinds <数据类型>→void|char|int|float|double|long

4,D->letter|digit <标识符>→<字符串>|<数字> //zwx：实际意义并非标识符

5,G->,DG|=DG|ε

6,F->+TF|-TF|*TF|\TF|ε

7,E->CDG|ε <形参>→<数据类型> <标识符><G>|ε

8,J->{I} <复合语句>→{ <语句列表>}

9,I->KI|DLI|MI|NI|QI|ε <语句列表>→<声明语句><语句列表>|<赋值语句|单运算符语句><语句列表>|<条件语句><语句列表>|<循环语句><语句列表>|<

跳转语句><复合语句>|ε

10,Q->break;|continue;|return R; <跳转语句>→ break;|continue;|return <表达式>;

11,K->CDG;<声明语句>→<数据类型><标识符>;

12,L->=R;|X;<赋值|单运算符语句>→<标识符>=<表达式>;

13,R->TF <表达式>

14,T->letter|digit F→表达式|标识符|数字|字符串

15,M->if(U){I}else{I} <条件语句>→if( <判断语句>)<{语句列表}> else <{语句列表}>//zwx：简化了条件语句

16,U->DVD <判断语句>-><标识符><关系运算符><临时变量>

17,V-> >|<|==|<=|>=|!= <关系运算符>→>|<|==|<=|>=|!=

18,N-> O|P <循环语句>→<for 语句>|<while 语句>//zwx：不考虑 do-while

19,O->for(DL;U;DX){I} <for 语句>→for(赋值语句|声明语句;判断语句; <标识符><D>){语句列表}

20,X->++|-- <自加自减>→++|--

21,P->while(U){I} <while 语句>→while(判断语句){语句列表}

为方便编程，定义 kinds = int | long | short | float | double | char

**2.2 根据产生式求 FIRST 集如下：**

表 1 FIRST 集

| 标识符 | 非终结符 | FIRST 集 |
|---|---|---|
| A | 源程序 | kinds |
| B | 函数体 | kinds |
| C | 数据类型 | kinds |

| 标识符 | 非终结符 | FIRST 集 |
|---|---|---|
| D | 标识符 | letter,digit |
| E | 形参 | kinds，',' |
| G | G | ,，= |
| I | 语句列表 | {, ε,letter,if,for,whilebreak,continue,return |
| J | 复合语句 | { |
| K | 声明语句 | kinds |
| L | 赋值语句 | letter |
| M | 条件语句 | if |
| N | 循环语句 | for,while |
| O | for 语句 | for |
| P | while 语句 | while |
| Q | 跳转语句 | break,continue,return |
| R | 表达式 | letter，digit |
| T | T | letter，digit |
| F | F | +,-,*,/ |
| U | 判断语句 | letter |
| V | 关系运算符 | <,>,!=,>=,<=,== |
| X | 自加自减 | ++,-- |

## 2.3 求 FOLLOW 集如下：

表 2 FOLLOW 集

| 标识符 | 非终结符 | FOLLOW 集 |
|---|---|---|
| A | 源程序 | # |
| B | 函数体 | # |
| C | 数据类型 | letter,digits |
| D | 标识符 | letter,digits,=,(,<,>,!=,>=,<=,==,\|\|,&&,!,;,++,-- |
| E | 形参 | , ) |
| G | G | |

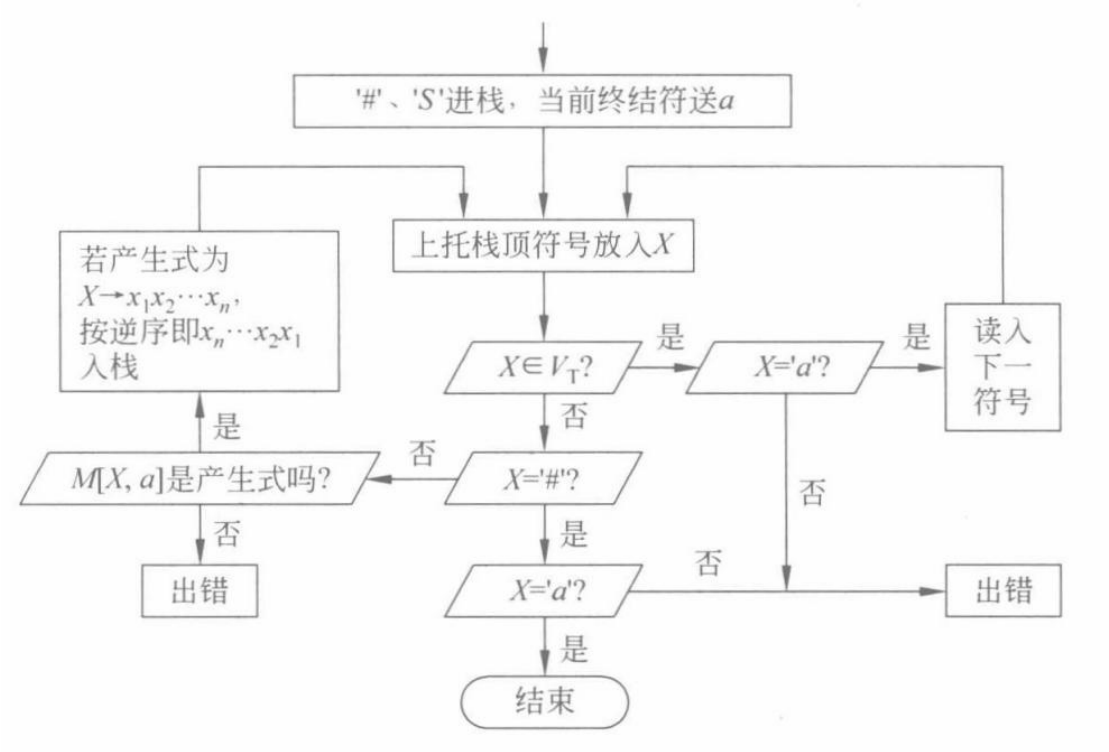
| 标识符 | 非终结符 | FOLLOW 集 |
| --- | --- | --- |
| H | 数字//zwx：同上 | kinds,letter,digits,=,(,<,>,!=,>=,<=,==,\|\|,&&,! |
| I | 语句列表 | } |
| J | 复合语句 | # |
| K | 声明语句 | {,ε,letter,if,for,while,break,continue,return,} |
| L | 赋值语句 | {,ε,letter,if,for,while,break,continue,return,} |
| M | 条件语句 | {,ε,letter,if,for,while,break,continue,return,} |
| N | 循环语句 | {,ε,letter,if,for,while,break,continue,return,} |
| O | for 语句 | {,ε,letter,if,for,while,break,continue,return,} |
| P | while 语句 | {,ε,letter,if,for,while,break,continue,return,} |
| Q | 跳转语句 | {,ε,letter,if,for,while,break,continue,return,} |
| R | 表达式 | +,-,*,/ |
| T | T | +,-,*,/ |
| F | F | *,/,+,- |
| U | 判断语句 | ) |
| V | 关系运算符 | digit,letter |
| X | 自加自减 | ) |

## 2.4 求得的预测分析表如下：（仅截图展示部分，详细见 gitee 仓库）

| | kinds:( | 0:digit | 1:lette | 5:( | 6:) | 9:{ | 10:} | 11:, | 12:; | 47:if | 54:for | 53:whi | 56:bre |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A:100 | A->B | | | | | | | | | | | | |
| B:101 | B->CD(E)J | | | | | | | | | | | | |
| C:102 | C->kinds | | | | | | | | | | | | |
| D:103 | | D->dig | D->letter | | | | | | | | | | |
| E:104 | E->CDG | | | | G->ε | | | | | | | | |
| F:105 | | | | | | | | | F->ε | | | | |
| G:106 | | | | | G->ε | | | G->,DC | G->ε | | | | |
| I:108 | I->KI | | I->DLI | | | | I->ε | | | I->MI | I->NI | I->NI | I->QI |
| J:109 | | | | | | J->{I} | | | | | | | |
| K:110 | K->CDG; | | | | | | | | | | | | |
| L:111 | | | | | | | | | | | | | |
| M:112 | | | | | | | | | | M->if(U){I}else{I} | | | |
| N:113 | | | | | | | | | | | N->O | N->P | |
| O:114 | | | | | | | | | | | O->for(DL U;DX){I} | | |
| P:115 | | | | | | | | | | | | P->while(U){I} | |
| Q:116 | | | | | | | | | | | | | Q->bre |
| R:117 | | R->TF | R->TF | | | | | | | | | | |
| T:119 | | T->digi | T->letter | | | | | | | | | | |
| U:120 | | | U->DVD | | | | | | | | | | |
| V:121 | | | | | | | | | | | | | |
| X:123 | | | | | | | | | | | | | |

## 3.语法分析程序的算法思想

### 3.1 程序流程图




### 3.2 根据预测分析表编制程序结果如下：

输入数据为实验一报告中词法分析器处理后的结果，输出结果内容太多，只截取开头和结尾部分展示： 开头：

```
步骤   符号栈                  当前输入的符号    动作
   0 #A                    int
   1 #B                    int             A->B
   2 #J)E(DC               int             B->CD(E)J
   3 #J)E(Dkinds           int             C->kinds
   4 #J)E(D                main
   5 #J)E(letter           main            D->letter
   6 #J)E(                 (
   7 #J)E                  )
   8 #J)                   )               E->ε
   9 #J                    {
  10 #}I{                  {               J->{I}
  11 #}I                   int
  12 #}IK                  int             I->KI
  13 #}I;GDC               int             K->CDG;
  14 #}I;GDkinds           int             C->kinds
  15 #}I;GD                x
  16 #}I;Gletter           x               D->letter
  17 #}I;G                 ,
  18 #}I;GD,               ,               G->,DG
  19 #}I;GD                y
  20 #}I;Gletter           y               D->letter
  21 #}I;G                 ;
  22 #}I;                  ;               G->ε
  23 #}I                   char
  24 #}IK                  char            I->KI
  25 #}I;GDC               char            K->CDG;
  26 #}I;GDkinds           char            C->kinds
  27 #}I;GD                ch
  28 #}I;Gletter           ch              D->letter
  29 #}I;G                 ;
  30 #}I;                  ;               G->ε
行 25 列 40                                          字符数 5686
```

结尾：

```
 99 #}I}I;R=            =             L->=R;
100 #}I}I;R             e
101 #}I}I;FT            e             R->TF
102 #}I}I;Fletter       e             T->letter
103 #}I}I;F             +
104 #}I}I;FT+           +             F->+TF
105 #}I}I;FT            b
106 #}I}I;Fletter       b             T->letter
107 #}I}I;F             ;
108 #}I}I;              ;             F->ε
109 #}I}I               b
110 #}I}ILD             b             I->DLI
111 #}I}ILletter        b             D->letter
112 #}I}IL              ++
113 #}I}I;X             ++            L->X;
114 #}I}I;++            ++            X->++
115 #}I}I;              ;
116 #}I}I               }
117 #}I}                }             I->ε
118 #}I                 return
119 #}IQ                return        I->QI
120 #}I;Rreturn         return        Q->returnR;
121 #}I;R               binary:0
122 #}I;FT              binary:0      R->TF
123 #}I;Fdigit          binary:0      T->digit
124 #}I;F               ;
125 #}I;                ;             F->ε
126 #}I                 }
127 #}                  }             I->ε
128 #                   #
succeess!
```

行 18 列 49　　字符数 5686

### 3.3 算法思想及总结：

（1）分析实验运行结果，对实验一中所示输入分析正确无误，符合题目要求，本实验程序可扩展性强，自定义文法可扩展性强，经过测试，符合自定义文法分析要求。

（2）由于自定义文法较为复杂，上述 first 集和 follow 集仍存在错误，预测分析表对于测试数据符合要求，但仍可能存在问题，本次实验所有文件我都已在 gitee 平台开源，后续经过改正优化后也会更新到我的仓库。

（3）文法分析器的难点主要在于预测分析表的求解，本次实验通过手动计算，实现过程复杂，程序实现流程图如课本图 4.4 所示，预测分析表通过二维整形数组嵌套动态字符串数组实现，上述文法中终结符和非终结符在程序逻辑底层都映射为整形数值，方便程序扩展和优化修改。

## 4. 语法分析程序的 C 语言程序框架

```
#include<bits/stdc++.h>
using namespace std;
//grammer analyzer of zwx
//暂时没加出错行数
string
sign[100]={"digit","letter","#","<",">","(",")","[","]","{","}",",",";","+","-","*","/","%","!","'","\"","=","++","--","&
&","||",">=","<=","==","!=",">>","<<",":=","int","long","short","float","double","char","unsigned","signed",
"const","void","volatile","enum","struct","union","if","else","goto","switch","case","do","while","for","co
ntinue","break","return","default","typedef","auto","register","extern","static","sizeof","include","
","kinds"};
stack<int>sta;
stack<int>atmsta;
int no=0;//行数
char str[20];
int atm;
vector<int> predit[200][100];//预测分析表
void init(){//预测分析表的读入
    FILE *fp=freopen("lexinit.txt","r",stdin);
    int x,y, n,t;
    string str;
    for(int k=0;k<51;k++)
    {
    scanf("%d %d %d",&x,&y,&n);
        for(int i=0;i<n;i++){
            cin>>str;
            if(str[0]>='A'&&str[0]<='Z')t=(str[0]-'A'+100);
            else if(str=="kinds")t=67;
            else if(str=="ε")t=-1;
            else {
                for(t=0;t<70;t++){
                    if(sign[t]==str)break;
                }
            }
            //cout<<t<<endl;
            predit[x][y].push_back(t);
        }
    }
//      x=114 ;y=54;
//      for(int i=0;i<predit[x][y].size();i++)
//      cout<<predit[x][y][i]<<" ";
  //    cout<<predit[123][22].size()<<endl;
```

```
	fclose(fp);
}
void print(int x ,int y){
		//print
		int len=0;//栈中元素长度，为了美化输出
		printf("%4d ",no);
		while(sta.size()){
			atmsta.push(sta.top());
			sta.pop();
		}
		len=0;
		while(atmsta.size()){
			if(atmsta.top()==-1){
				printf("$");len++;
			}
			else if(atmsta.top()>=100) {
				printf("%c",(char)(atmsta.top()-100+'A'));
				len++;
			}
			else {
				cout<<sign[atmsta.top()];
				len+=sign[atmsta.top()].size();
			}
			sta.push(atmsta.top());
			atmsta.pop();
		}
		for(int i=len;i<20;i++)printf(" ");
			printf("%s",str);
		for(int i=strlen(str);i<14;i++)printf(" ");
		if(x>=0&&y>=0){
			cout<<(char)(x-100+'A')<<"->";
			for(unsigned int i=0;i<predit[x][y].size();i++)
				{
					int t =predit[x][y][i];
					if(t<=67&&t>=0)cout<<sign[t];
					else if(t<0)cout<<"ε";
					else cout<<(char)(t-100+'A');
				}
		}
		printf("\n");
}
void analyzer(){
	//init
	sta.push(2);
```

```
sta.push(100);
printf("步骤    符号栈                    当前输入的符号      动作\n");
bool f=true;
while(scanf("< %d ,%s >",&atm,str)){
if(!f)break;
    getchar();
    //if(atm==2)break;  代码中有#的问题还没有解决
    //printf("%d %s\n",atm,str);
    print(-1,-1);no++;
    //action
    if(atm>=33&&atm<=45)atm=67;
    while(true){
    int now=sta.top();sta.pop();
    if(now<=67&&now!=2){//是一个终结符
        if(now==atm)break;
        else {
            printf("error!");
            f=false;
            break;
        }
    }
    else if(now==2){
        if(now==2&&atm==2&&sta.size()==0){
            printf("succeess!");
            break;
        }
        else {
            printf("error!");
            f=false;
            break;
        }
    }
    else {
    int ans;
    //cout<<now<<" "<<atm<<" ";
        if(predit[now][atm].size()){
            for(int i=predit[now][atm].size()-1;i>=0;i--)
                {
                    ans=predit[now][atm][i];
                    //cout<<ans<<" ";
                    if(ans>=0&&ans<=200)sta.push(ans);
                }
            //cout<<endl;
            print(now,atm);
```

```
                    no++;
                }
                else {
                    printf("error!");
                    f=false;
                    break;
                }
            }
            }
            if(atm==2)break;
        }
}
int main (){
    init();
    freopen("lexicalResult.txt","r",stdin);//文件标准读入
    freopen("result.txt","w",stdout);//文件标准输出
    analyzer();
    return 0;
}
```