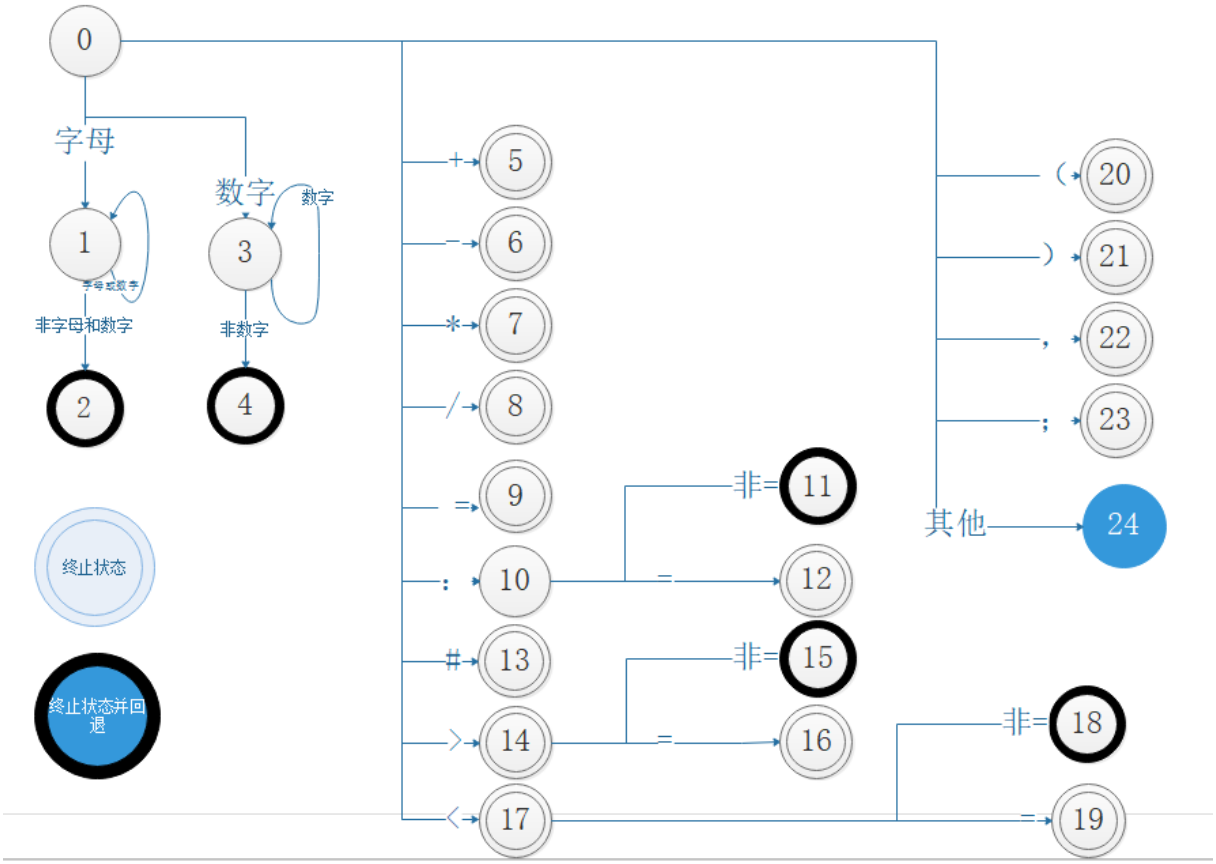
# 计算机学院编译原理与技术课程实验报告

| 实验题目：词法分析 | | 学号：201720130148 |
|---|---|---|
| 日期：2020/4/19 | 班级： 17.2 | 姓名：陈加乐 |
| Email：1345068110@qq.com | | |

实验步骤与内容：

首先给出词法分析的状态图如下：




编制单词编码表如下

| 单词符号 | 种别编码 | 助忆符 | 内码值 |
|---|---|---|---|
| const | 1 | NULL | NULL |
| var | 2 | NULL | NULL |
| procedure | 3 | NULL | NULL |
| begin | 4 | NULL | NULL |
| end | 5 | NULL | NULL |
| odd | 6 | NULL | NULL |
| if | 7 | NULL | NULL |
| then | 8 | NULL | NULL |

| | | | |
|---|---|---|---|
| call | 9 | NULL | NULL |
| while | 10 | NULL | NULL |
| do | 11 | NULL | NULL |
| read | 12 | NULL | NULL |
| write | 13 | NULL | NULL |
| = | 14 | NULL | NULL |
| := | 15 | NULL | NULL |
| + | 16 | NULL | NULL |
| - | 17 | NULL | NULL |
| * | 18 | NULL | NULL |
| / | 19 | NULL | NULL |
| # | 20 | NULL | NULL |
| < | 21 | NULL | NULL |
| <= | 22 | NULL | NULL |
| > | 23 | NULL | NULL |
| >= | 24 | NULL | NULL |
| , | 25 | NULL | NULL |
| ; | 26 | NULL | NULL |
| ( | 27 | NULL | NULL |
| ) | 28 | NULL | NULL |
| identifier | 29 | NULL | NULL |
| constant | 30 | NULL | NULL |

首先一些必要的准备：

例如，建立保留字表，和单词类别表

```
//保留字表
map<string,int> reserve;
map<int,string> kw;//单词种别及其对应的单词 符号
```

```
// 创建保留字表reserve(该表同样兼具单词编码表)，还有编码对应单词表kw
void create_keytable(){
//保留字表 ，也是保留字的单词编码表，
//            单词编码还有另外的两类单词，标识符和常量，
//                        其类别分别使用identifier ,constant
    for(int i=0;i<kwsize;i++)
        reserve[keyword[i]]=i+1;
//单词符号表
//  关键字一字一类
//标识符统归一类
//常数，统归一类；（可按类型，整型，实型，布尔型
    for(int i=1;i<=kwsize;i++)
        kw[i]=keyword[i-1];
    kw[identifier]="identifier";
    kw[constant]="constant";
}
```

还有一些有用的函数

```
//string数据转换为int型
int strtoi(string x){
```

```
//  预处理子程序，删除多余空格，tab，回车，换行、注释
string preprocessing(string s)
```

```
//错误处理
void werror(){

//输出显示单词类别表
void show word coding table(){

//屏幕输出单词序列
void show(seq *head){
```

（1）首先读取 PL0 源程序：建立函数 fr（）参数为源程序路径名

```
//读取PL0语言源程序
string fr(char *name){
    FILE *fp;
    if ((fp = fopen(name, "r")) == NULL) {
        printf("couldn't open input file %s\n", name);
        return 0;
    }
    fseek(fp, 0, 2);          //定位指针到尾部
    int fileLength = ftell(fp);
    fseek(fp, 0, 0);
    cout<<"file xlength:"<<fileLength<<endl;
    char *buff=new char[fileLength];
    int amount=fread(buff,sizeof(bool),fileLength,fp);
    return buff;
}
```

（2）对源程序形成的长字符串，根据上面的状态图进行词法分析

```
//词法分析器产生，单词符号的二元式序列
//参数是一个源程序长字符串
void lexer(string s){
    //构造DFA
    string c="";
    int i=0,len=s.length();
    while(i<len){
        while(s[i]==' '||s[i]=='\n')    i++;
        if(isalpha(s[i])){
            c+=s[i++];
            while(isalnum(s[i]))
                c+=s[i++];
        //  i--;//回退不需要
            gens(c,0,1);//存储单词
            c="";//c清空
        }else if(isdigit(s[i])){
            int val=0;
            val=val*10+s[i]-'0';
            c+=s[i++];
            while(isdigit(s[i]))
            {
                val=val*10+s[i]-'0';
                c+=s[i++];
            }
        //  i--;//回退 不需要
            gens(c,1,1);//存储单词
            c="";//c清空
        }else if(s[i]=='+'){
            c+=s[i++];  gens(c);    c="";
        }else if(s[i]=='-'){
            c+=s[i++];  gens(c);    c="";
```

（3）词法分析要产生单词序列的二元式

首先定义存储格式

```
//源程序经词法分析后产生的单词序列
struct seq{
    int k;//单词种别
    string v;//单词自身值
    seq* next;
    seq(){k=-1;v="";next=NULL;}
}*head,*tail;
```

然后，把词法分析程序，识别出的单词，存起来

```
//把识别出来的单词，存进源程序的单词序列中去
void gens(string s,bool v1=0,bool v2=0){
//  v1 =0, v2=0,关键字
//  v1 =0, v2=1 , 标识符
//  v1 =1, v2=1 , 常数
    if(!head )  tail=new seq;
    tail->next=new seq;
    if(!head )  head=tail,delete head,head=tail->next;
    tail=tail->next;
    if(reserve[s])  //关键字:
        tail->k=reserve[s];
    else if(!v1&&v2){//是标识符    要构建标识符的符号表
        tail->k=identifier;
        tail->v=s;
        //存进table表
        userword *x=new userword;//(s) k val level ADR
        x->k=identifier;
        x->level=level;
        table[s]=x;
    }
    else if(v1&&v2){//是常数 存进常数表
        tail->k=constant;
        tail->v=to_string(ci);
        //要使用常数的值时，需要把string数据转换为int型
        //存进num表
        num[ci++]=strtoi(s);
    }
}
```

（4）建立常量表 num 和标识符表 table

```
//建立常量值表num ,及下一个可以放置常量的地址
int num[maxnum],ci;

//要创建一个table表，存储源程序中出现的标识符
// 包括，常量名，变量名，过程名
//table的格式，name; kind; val(常量);level;ADR(变量);
//这里的ADR是存储常量实际值的指针，即num中的偏移量
struct userword{
//  string name;  使用map，省了这一项
    int k;  //单词类别
    int val; //常量值
    int level;//量的层次
    int ADR; //变量值的地址
};
map<string, userword*> table;
```

（5）目前为止，main 函数输出结果：

```
int main()
{
//创建保留字表reserve(该表同样兼具单词编码表)，还有编码对应单词表kw
    create_keytable();
    //读取PL0源程序文件
    string t=fr(name);
    cout<<t<<endl;
//   t=preprocessing(t);//对源程序做预处理
//  show_word_coding_table();//屏幕输出单词类别表
    //词法分析程序，产生单词序列链表，head
    lexer(t);
    //输出单词序列链表head
    show(head);
    return 0;
}
```

```
file xlength:171
const a=10;
var b,c;
procedure p;
begin
  c:=b+a
end;
begin
  read(b);
  while b#0 do
    begin
      call  p;
      write(2*c);
      read(b);
    end
end.

Program END
<       const   ,           1   ,           >
< identifier    ,          29   ,           a >
<           =   ,          14   ,           >
<    constant   ,          30   ,          10 >
<           ;   ,          26   ,           >
<         var   ,           2   ,           >
< identifier    ,          29   ,           b >
<           ,   ,          25   ,           >
< identifier    ,          29   ,           c >
<           ;   ,          26   ,           >
<   procedure   ,           3   ,           >
< identifier    ,          29   ,           p >
<           ;   ,          26   ,           >
<       begin   ,           4   ,           >
< identifier    ,          29   ,           c >
<          :=   ,          15   ,           >
< identifier    ,          29   ,           b >
<           +   ,          16   ,           >
< identifier    ,          29   ,           a >
<         end   ,           5   ,           >
<           ;   ,          26   ,           >
<       begin   ,           4   ,           >
<        read   ,          12   ,           >
<           (   ,          27   ,           >
< identifier    ,          29   ,           b >
<           )   ,          28   ,           >
<           ;   ,          26   ,           >
<       while   ,          10   ,           >
< identifier    ,          29   ,           b >
<           #   ,          20   ,           >
<    constant   ,          30   ,           0 >
<          do   ,          11   ,           >
<       begin   ,           4   ,           >
<        call   ,           9   ,           >
< identifier    ,          29   ,           p >
<           ;   ,          26   ,           >
<       write   ,          13   ,           >
<           (   ,          27   ,           >
<    constant   ,          30   ,           2 >
<           *   ,          18   ,           >
```

已完成代码：

```
#include<iostream>
#include<string>
#include<cstring>
#include<map>
#include<algorithm>
#include<stdio.h>
#include<iomanip>
#define ll long long
using namespace std;

#define kwsize 28
#define identifier 29
#define constant 30
//全局变量
const int maxnum=1000;//程序中出现的整型变量的最大个数
string
    keyword[]={
    //关键字 13
    "const","var","procedure","begin","end",
    "odd","if","then","call","while","do","read","write",
    //运算符 11
    "=",":=","+","-","*","/","#","<","<=",">",">=",
    //分界符 4
    ",",";","(",")"
    //  2
//  "identifier","constant"
    };
    //常量,
    //标识符     <标识符> → <字母>{<字母>|<数字>}
    /*
字母（不区分大小写）：isalpha();大写字母：isupper();
小写字母：islower();数字：isdigit();字母和数字：isalnum();
    */
ll len;
int i,vnum,//标识符数目
    level; //当前程序所处的层次
//保留字表
map<string,int> reserve;
map<int,string> kw;//单词种别及其对应的单词 符号
char
*name="E:\\My_project\\Lesson\\compiler_PTT\\lab\\PL0_code\\PL0_code1.in";

//源程序经词法分析后产生的单词序列
```

```
struct seq{
    int k;//单词种别
    string v;//单词自身值
    seq* next;
    seq(){k=-1;v="";next=NULL;}
}*head,*tail;
//建立常量值表 num ,及下一个可以放置常量的地址
int num[maxnum],ci;

//要创建一个 table 表，存储源程序中出现的标识符
// 包括，常量名，变量名，过程名
//table 的格式，name; kind; val(常量);level;ADR(变量);
//这里的 ADR 是存储常量实际值的指针，即 num 中的偏移量
struct userword{
//  string name;  使用 map，省了这一项
    int k;  //单词类别
    int val; //常量值
    int level;//量的层次
    int ADR; //变量值的地址
};
map<string, userword*> table;
//string 数据转换为 int 型
int strtoi(string x){
    if(x=="")  return -1;
    int s=0,l=x.length();
    for(int i=0;i<l;i++)
        s=s*10+x[i]-'0';
    return s;
}


// 创建保留字表 reserve(该表同样兼具单词编码表)，还有编码对应单词表 kw
void create_keytable(){
//保留字表  ,也是保留字的单词编码表,
//            单词编码还有另外的两类单词，标识符和常量,
//                    其类别分别使用 identifier ,constant
    for(int i=0;i<kwsize;i++)
        reserve[keyword[i]]=i+1;
//单词符号表
//  关键字一字一类
//标识符统归一类
//常数，统归一类；(可按类型，整型，实型，布尔型
    for(int i=1;i<=kwsize;i++)
        kw[i]=keyword[i-1];
```

```
    kw[identifier]="identifier";
    kw[constant]="constant";
}

//读取 PL0 语言源程序
string fr(char *name){
    FILE *fp;
    if ((fp = fopen(name, "r")) == NULL) {
        printf("couldn't open input file %s\n", name);
        return 0;
    }
    fseek(fp, 0, 2);      //定位指针到尾部
     int fileLength = ftell(fp);
     fseek(fp, 0, 0);
     cout<<"file xlength:"<<fileLength<<endl;
     char *buff=new char[fileLength];
     int amount=fread(buff,sizeof(bool),fileLength,fp);
     return buff;
}

//  预处理子程序，剔除多余空格，tab，回车，换行、注释
string preprocessing(string s)
{
    int index = 0;
     if( !s.empty())
     {//这里不对，不是去除全部空格，而是去除多余空格
         while( (index = s.find(' ',index)) != string::npos)
         {
             s.erase(index,1);
         }
         index=0;
         while( (index = s.find('\n',index)) != string::npos)
         {
             s.erase(index,1);
         }
     }
    return s;
}

//错误处理
void werror(){
    cout<<"ERROR! \n";
}
```

```
//输出显示单词类别表
void show_word_coding_table(){
    for(int i=0;i<kwsize;i++)
        cout<<keyword[i]<<" "<< reserve[keyword[i]]<<endl;
    cout<<"identifier "<<identifier<<endl;
    cout<<"constant "<<constant<<endl;
}

//屏幕输出单词序列
void show(seq *head){
    seq* cur=head;
    while(cur){
        cout<<"< ";
        cout.width(10);  cout<<kw[cur->k]<<"   ,   ";
        cout.width(5);  cout<<cur->k<<",   ";
        cout.width(5);
            if(cur->k==constant&&strtoi(cur->v)!=-1)
    cout<<num[strtoi(cur->v)] <<" >\n";
            else cout<<cur->v <<" >\n";
        cur=cur->next;
    }
}


//把识别出来的单词，存进源程序的单词序列中去
void gens(string s,bool v1=0,bool v2=0){
//  v1 =0, v2=0,关键字
//  v1 =0, v2=1 ，标识符
//  v1 =1, v2=1 ，常数
    if(!head ) tail=new seq;
    tail->next=new seq;
    if(!head ) head=tail,delete head,head=tail->next;
    tail=tail->next;
    if(reserve[s])    //关键字：
        tail->k=reserve[s];
    else if(!v1&&v2){//是标识符    要构建标识符的符号表
        tail->k=identifier;
        tail->v=s;
        //存进 table 表
        userword *x=new userword;//(s) k val level ADR
        x->k=identifier;
        x->level=level;
        table[s]=x;
    }
```

```
    else if(v1&&v2){//是常数 存进常数表
        tail->k=constant;
        tail->v=to_string(ci);
        //要使用常数的值时，需要把 string 数据转换为 int 型
        //存进 num 表
        num[ci++]=strtoi(s);
    }
}


//词法分析器产生，单词符号的二元式序列
//参数是一个源程序长字符串
void lexer(string s){
    //构建 DFA
    string c="";
    int i=0,len=s.length();
    while(i<len){
        while(s[i]==' '||s[i]=='\n')    i++;
        if(isalpha(s[i])){
            c+=s[i++];
            while(isalnum(s[i]))
                c+=s[i++];
        //  i--;//回退不需要
            gens(c,0,1);//存储单词
            c="";//c 清空
        }else if(isdigit(s[i])){
            int val=0;
            val=val*10+s[i]-'0';
            c+=s[i++];
            while(isdigit(s[i]))
            {
                val=val*10+s[i]-'0';
                c+=s[i++];
            }
        //  i--;//回退 不需要
            gens(c,1,1);//存储单词
            c="";//c 清空
        }else if(s[i]=='+'){
            c+=s[i++];gens(c);   c="";
        }else if(s[i]=='-'){
            c+=s[i++];gens(c);   c="";
        }else if(s[i]=='*'){
            c+=s[i++];gens(c);   c="";
        }else if(s[i]=='/'){
```

```
            c+=s[i++];gens(c);   c="";
        }else if(s[i]=='='){
            c+=s[i++];gens(c);   c="";
        }else if(s[i]==':'){
            c+=s[i++];
            if(s[i]=='='){
                c+=s[i++];
                gens(c);
                c="";
            }else werror();
        }else if(s[i]=='#'){
            c+=s[i++];gens(c);   c="";
        }else if(s[i]=='>'){
            c+=s[i++];
            if(s[i]=='=')
                c+=s[i++];
            gens(c);
            c="";
        }else if(s[i]=='<'){
            c+=s[i++];
            if(s[i]=='=')
                c+=s[i++];
            gens(c);
            c="";
        }else if(s[i]=='('){
            c+=s[i++];gens(c);   c="";
        }else if(s[i]==')'){
            c+=s[i++];gens(c);   c="";
        }else if(s[i]==','){
            c+=s[i++];gens(c);   c="";
        }else if(s[i]==';'){
            c+=s[i++];gens(c);   c="";
        }else if(s[i++]=='.'){
            cout<<"\nProgram END\n";
        }else werror();
    }

}


int main()
{
//创建保留字表 reserve(该表同样兼具单词编码表)，还有编码对应单词表 kw
    create_keytable();
```

```
    //读取 PL0 源程序文件
    string t=fr(name);
    cout<<t<<endl;
//   t=preprocessing(t);//对源程序做预处理
//  show_word_coding_table();//屏幕输出单词类别表
    //词法分析程序，产生单词序列链表，head
    lexer(t);
    //输出单词序列链表 head
    show(head);
    return 0;
}
```