花了一天写的语法分析器的LR(1)分析表

                  LR(1)分析表生成器（FIRST需手动求...）

文法放在“wenfa.txt”文件中，FIRST集放“first.txt”中

文法须为增广文法，每行一个产生式，“->”用“#”代替，非终结符用单个大写字母，终结符用单个小写字母

FIRST集格式为“非终结符 终结符集合字符串”每行一条 如“T ni(” 空用“%”代替

生成ACTION表在action.txt中，GOTO表在goto.txt中

                                                                       -----by Decly     

#include<iostream>  
#include<vector>  
#include<string>  
#include<fstream>  
#include<cctype>  
#include<algorithm>  
#include<set>  
#include<map>  
#include<deque>  
using namespace std;  
class a\_xiang  
{  
public:  
 a\_xiang::a\_xiang(){}  
 a\_xiang::a\_xiang(const string& str,set<char>& forw):a\_shizi(str),forword(forw){}  
 string& geta\_shizi(){return a\_shizi;}  
 set<char>& getforword(){return forword;}  
 void push\_forword(set<char>& sv){forword.insert(sv.begin(),sv.end());}  
 friend bool operator==(const a\_xiang& lhs,const a\_xiang& rhs);  
private:  
 string a\_shizi;  
 set<char> forword;  
};  
class xiangji  
{  
public:  
 void closure();  
 void push\_a\_xiang(const a\_xiang orig){xiang.push\_back(orig);}  
 void output();  
 vector<a\_xiang>& get\_xiang(){return xiang;}  
 set<pair<char,int> >& get\_action(){return action;}  
 friend bool operator==(const xiangji& lhs,const xiangji& rhs);  
private:  
 vector<a\_xiang> xiang;  
 set<pair<char,int> > action;  
};

map<char,set<char> > first;  
vector<string> wenfa,wenfa\_dian;  
deque<xiangji> gototu;  
set<char> zhongjiefu,feizhongjiefu;  
vector<vector<int> >actions,gotos;  
bool operator==(const a\_xiang& lhs,const a\_xiang& rhs)  
{  
 if(lhs.a\_shizi!=rhs.a\_shizi) return false;  
 if(lhs.forword.size()!=rhs.forword.size()) return false;  
 set<char>::const\_iterator lhs\_it=lhs.forword.begin(),rhs\_it=rhs.forword.begin();  
 while(lhs\_it!=lhs.forword.end())  
 {  
  if(\*lhs\_it!=\*rhs\_it) return false;  
  ++lhs\_it;  
  ++rhs\_it;  
 }  
 return true;  
}  
bool operator==(const xiangji& lhs,const xiangji& rhs)  
{  
 if(lhs.xiang.size()==rhs.xiang.size())  
 {  
  for(vector<a\_xiang>::const\_iterator lhs\_it=lhs.xiang.begin();lhs\_it!=lhs.xiang.end();++lhs\_it)  
  {  
   vector<a\_xiang>::const\_iterator rhs\_it=rhs.xiang.begin();  
   for(;rhs\_it!=rhs.xiang.end();++rhs\_it)  
    if(\*lhs\_it==\*rhs\_it) break;  
   if(rhs\_it==rhs.xiang.end()) return false;  
  }  
  return true;  
 }  
 else return false;  
}  
void changewenfa()  
{  
 vector<string>::iterator it=wenfa.begin();  
 while(it!=wenfa.end())  
 {  
  string a\_line(\*it);  
  if(a\_line.size()==2) a\_line.push\_back('!');  
  else  
  {  
   string::iterator p=a\_line.begin();  
   ++p;++p;  
   a\_line.insert(p,'!');  
  }  
  wenfa\_dian.push\_back(a\_line);  
  ++it;  
 }  
}  
void inputwenfa(const string& wenfa\_file)//读入文法文件  
{  
 ifstream in;  
 in.open(wenfa\_file.c\_str());  
 string a\_line;  
 while(getline(in,a\_line))  
  wenfa.push\_back(a\_line);  
 in.close();  
 changewenfa();  
}

void inputfirst(const string& first\_file)//读入FIRST文件  
{  
 ifstream in;  
 in.open(first\_file.c\_str());  
 string a\_line;  
 while(getline(in,a\_line))  
 {  
  char ch=a\_line[0];  
  for(int i=2;i!=a\_line.size();++i)  
   first[ch].insert(a\_line[i]);  
 }  
 in.close();  
}

bool isfeizhongjiefu(char c)  
{  
 if(c>='A'&&c<='Z') return true;  
 else return false;  
}  
bool iszhongjiefu(char c)  
{  
 return !isfeizhongjiefu(c);  
}  
vector<string> chanshengshi(char c)  
{  
 vector<string> vec;  
 for(vector<string>::iterator it=wenfa\_dian.begin();it!=wenfa\_dian.end();++it)  
  if((\*it)[0]==c) vec.push\_back(\*it);  
 return vec;  
}  
set<char> findfirst(const string& str,set<char>& forword)  
{  
 if(str=="") return forword;  
 else if(iszhongjiefu(str[0]))  
 {  
  set<char> setc;  
  setc.insert(str[0]);  
  return setc;  
 }  
 else// if(isfeizhongjiefu(str[0]))  
 {  
  set<char> setc;  
  string::const\_iterator p=str.begin();  
  while(p!=str.end())  
  {  
   setc.insert((first[\*p]).begin(),(first[\*p]).end());  
   if((first[\*p]).find('%')!=first[\*p].end()) ++p;  
   else break;  
  }  
  if(p==str.end()) setc.insert(forword.begin(),forword.end());  
  setc.erase('%');  
  return setc;  
 }  
}

void xiangji::closure()  
{  
 for(vector<a\_xiang>::size\_type it=0;it!=xiang.size();++it)  
 {  
  for(string::iterator p=xiang[it].geta\_shizi().begin();p!=xiang[it].geta\_shizi().end();++p)  
  {  
   if(\*p=='!')  
   {  
    if((++p)!=xiang[it].geta\_shizi().end()&&isfeizhongjiefu(\*p))  
    {  
     string hou(++p,xiang[it].geta\_shizi().end());  
       
     set<char> forword\_2(findfirst(hou,xiang[it].getforword()));  
     vector<string> shi(chanshengshi(\*(--p)));  
     for(vector<string>::iterator jp=shi.begin();jp!=shi.end();++jp)  
     {  
      vector<a\_xiang>::iterator xiang\_it=xiang.begin();  
      while(xiang\_it!=xiang.end())  
      {  
       if((\*jp)==xiang\_it->geta\_shizi())  
       {  
        xiang\_it->push\_forword(forword\_2);  
        break;  
       }  
       ++xiang\_it;  
      }  
      if(xiang\_it==xiang.end())  
       xiang.push\_back(a\_xiang((\*jp),forword\_2));  
     }  
     break;  
    }  
    --p;  
   }  
  }  
    
 }  
}

void xiangji::output()  
{  
 vector<a\_xiang>::iterator ip=xiang.begin();  
 while(ip!=xiang.end())  
 {  
  cout<<(\*ip).geta\_shizi()<<'\t';  
  set<char>::iterator iit=ip->getforword().begin();  
  while(iit!=ip->getforword().end())  
  {  
   cout<<\*iit<<" ";  
   ++iit;  
  }  
  cout<<endl;  
  ++ip;  
 }  
}  
xiangji go\_to(xiangji& orig,char x)  
{  
 xiangji xia\_j;  
 for(vector<a\_xiang>::iterator it=orig.get\_xiang().begin();it!=orig.get\_xiang().end();++it)  
 {  
  string shizi(it->geta\_shizi());  
  string::iterator p=find(shizi.begin(),shizi.end(),'!');  
  if(++p!=shizi.end()&&(\*p)==x)  
  {  
   p=shizi.erase(p);  
   --p;  
   shizi.insert(p,x);  
   xia\_j.get\_xiang().push\_back(a\_xiang(shizi,it->getforword()));  
  }  
 }  
 xia\_j.closure();  
 return xia\_j;  
}  
bool isdifferent(xiangji& orig)  
{  
 bool flag=true;  
 for(vector<xiangji>::size\_type i=0;i!=gototu.size();++i)  
 {  
  if(orig==gototu[i])  
  {  
   flag=false;  
   break;  
  }  
 }  
 return flag;  
}

void items()  
{  
 xiangji start;  
 set<char> start\_forword;  
 start\_forword.insert('$');  
 start.push\_a\_xiang(a\_xiang(wenfa\_dian.front(),start\_forword));  
 start.closure();  
 gototu.push\_back(start);  
 for(vector<xiangji>::size\_type i=0;i!=gototu.size();++i)  
 {  
  for(vector<a\_xiang>::iterator p=gototu[i].get\_xiang().begin();p!=gototu[i].get\_xiang().end();++p)  
  {  
   string::iterator it=find(p->geta\_shizi().begin(),p->geta\_shizi().end(),'!');  
   ++it;  
   if(it==p->geta\_shizi().end()) continue;  
   xiangji a\_new\_xiangji(go\_to(gototu[i],\*it));  
   if(isdifferent(a\_new\_xiangji))  
   {  
    gototu[i].get\_action().insert(make\_pair(\*it,gototu.size()));  
    gototu.push\_back(a\_new\_xiangji);  
   }  
   else  
   {  
    for(vector<xiangji>::size\_type ip=0;ip!=gototu.size();++ip)  
    {  
     if(a\_new\_xiangji==gototu[ip])  
      gototu[i].get\_action().insert(make\_pair(\*it,ip));  
    }  
   }  
  }  
 }  
}  
void findzhongjiefu()  
{  
 for(vector<string>::iterator it=wenfa.begin();it!=wenfa.end();++it)  
 {  
  for(string::iterator p=it->begin();p!=it->end();++p)  
  {  
   if(isupper(\*p)) feizhongjiefu.insert(\*p);  
   else if(\*p!='#') zhongjiefu.insert(\*p);  
  }  
 }  
 zhongjiefu.insert('$');  
 feizhongjiefu.erase(wenfa[0][0]);  
}  
void creat\_goto()  
{  
 for(deque<xiangji>::iterator it=gototu.begin();it!=gototu.end();++it)  
 {  
  vector<int> line\_goto;  
  for(set<char>::iterator zhong\_it=feizhongjiefu.begin();zhong\_it!=feizhongjiefu.end();++zhong\_it)  
  {  
   set<pair<char,int> >::iterator p=it->get\_action().begin();  
   for(;p!=it->get\_action().end();++p)  
   {  
    if(\*zhong\_it==p->first)  
    {  
     line\_goto.push\_back(p->second);  
     break;  
    }  
   }  
   if(p==it->get\_action().end()) line\_goto.push\_back(0);  
  }  
  gotos.push\_back(line\_goto);  
 }  
}  
void creat\_action()  
{  
 for(deque<xiangji>::iterator it=gototu.begin();it!=gototu.end();++it)  
 {  
  vector<int> line\_action;  
  for(set<char>::iterator zhong\_it=zhongjiefu.begin();zhong\_it!=zhongjiefu.end();++zhong\_it)  
  {  
   set<pair<char,int> >::iterator p=it->get\_action().begin();  
   for(;p!=it->get\_action().end();++p)  
   {  
    if(\*zhong\_it==p->first)  
    {  
     line\_action.push\_back(p->second);  
     break;  
    }  
   }  
   if(p==it->get\_action().end()) line\_action.push\_back(0);  
  }  
  for(vector<a\_xiang>::iterator it\_shi=it->get\_xiang().begin();it\_shi!=it->get\_xiang().end();++it\_shi)  
  {  
   string str(it\_shi->geta\_shizi());  
   if(str[str.size()-1]=='!')  
   {  
    string::iterator ip\_str=str.end();  
    --ip\_str;  
    str.erase(ip\_str);  
    int num\_shi=0;  
    for(;num\_shi!=wenfa.size();++num\_shi)  
    {  
     if(str==wenfa[num\_shi]) break;  
    }  
    for(set<char>::iterator set\_it=it\_shi->getforword().begin();set\_it!=it\_shi->getforword().end();++set\_it)  
    {  
     int num=0;  
     for(set<char>::iterator it\_zhong=zhongjiefu.begin();it\_zhong!=zhongjiefu.end();++it\_zhong)  
     {   
      if(\*set\_it==\*it\_zhong) break;  
      ++num;  
     }  
     if(str[0]==wenfa[0][0]) line\_action[num]=999;  
     else line\_action[num]=-num\_shi;  
    }  
   }  
  }  
  actions.push\_back(line\_action);  
 }  
}  
void output\_fenxibiao()  
{  
 ofstream out\_action("action.txt"),out\_goto("goto.txt");  
 cout<<endl;  
 cout<<"LR(1)分析表："<<endl;  
 cout<<endl<<'\t';  
 set<char>::iterator zhong\_it=zhongjiefu.begin();  
 while(zhong\_it!=zhongjiefu.end())  
 {  
  cout<<\*zhong\_it++<<'\t';  
 }  
 set<char>::iterator feizhong\_it=feizhongjiefu.begin();  
 while(feizhong\_it!=feizhongjiefu.end())  
 {  
  cout<<\*feizhong\_it++<<'\t';  
 }  
 cout<<endl;  
 int line=0;  
 for(vector<vector<int> >::iterator i\_1=actions.begin(),i\_3=gotos.begin();i\_1!=actions.end();++i\_1,++i\_3,++line)  
 {  
  cout<<line<<'\t';  
  for(vector<int>::iterator i\_2=i\_1->begin();i\_2!=i\_1->end();++i\_2)  
  {  
   cout<<\*i\_2<<'\t';  
   out\_action<<\*i\_2<<',';  
  }  
  for(vector<int>::iterator i\_4=i\_3->begin();i\_4!=i\_3->end();++i\_4)  
  {  
   cout<<\*i\_4<<'\t';  
   out\_goto<<\*i\_4<<',';  
  }  
  cout<<endl;  
  out\_action<<endl;  
  out\_goto<<endl;  
 }

}  
void output\_zhuangtaitu()  
{  
 cout<<"有"<<gototu.size()<<"个状态"<<endl<<endl;  
 for(deque<xiangji>::size\_type itp=0;itp!=gototu.size();++itp)  
 {  
  cout<<"状态:"<<itp<<endl;  
  gototu[itp].output();  
  for(set<pair<char,int> >::iterator iiip=gototu[itp].get\_action().begin();iiip!=gototu[itp].get\_action().end();++iiip)  
  {  
   cout<<"输入："<<iiip->first<<"\t"<<"跳转到："<<iiip->second<<endl;  
  }  
  cout<<endl;  
 }  
}  
void output\_zhongjiefu()  
{  
 cout<<"终结符："<<zhongjiefu.size()<<"个"<<endl;  
 set<char>::iterator zhong\_it=zhongjiefu.begin();  
 while(zhong\_it!=zhongjiefu.end())  
 {  
  cout<<\*zhong\_it++<<' ';  
 }  
 cout<<endl;  
   
}  
void output\_feizhongjiefu()  
{  
 cout<<"非终结符："<<feizhongjiefu.size()<<"个"<<endl;  
 set<char>::iterator feizhong\_it=feizhongjiefu.begin();  
 while(feizhong\_it!=feizhongjiefu.end())  
 {  
  cout<<\*feizhong\_it++<<' ';  
 }  
 cout<<endl;  
}  
void output\_chanshengshi\_num()  
{  
 cout<<"规约时的产生式编号："<<endl;  
 int num=1;  
 vector<string>::iterator it=wenfa.begin();  
 ++it;  
 for(;it!=wenfa.end();++it,++num)  
 {  
  cout<<num<<'\t'<<\*it<<endl;  
 }  
}  
int main()  
{  
 inputwenfa("wenfa.txt");  
 inputfirst("first.txt");  
 findzhongjiefu();  
 items();  
 creat\_goto();  
 creat\_action();  
   
 output\_zhuangtaitu();  
 output\_zhongjiefu();  
 output\_feizhongjiefu();  
 output\_chanshengshi\_num();  
 output\_fenxibiao();  
   
  
  
 system("pause");  
 return 0;  
}

例：给定文法

          <program> ::= <block>

          <block> ::= <const-decl> <var-decl> <proc-decl> <statement>

          <const-decl> ::= const <const-assignment-list> ; | ε

          <const-assignment-list> ::= <ident> = <number>

            | <const-assignment-list> , <ident> = <number>

          <var-decl> ::= var <ident-list> ; |ε

          <ident-list> ::= <ident> | <ident-list> , <ident>

          <proc-decl> ::= <proc-decl> procedure <ident> ; <block> ; |ε

          <statement> ::= <ident> := <expression>

            | call <ident>

            | begin <statement-list> end

            | if <condition> then <statement>

            | while <condition> do <statement>

            |ε

          <statement-list> ::= <statement> | <statement-list> ; <statement>

          <condition> ::= odd <expression> | <expression> <relation> <expression>

          <relation> ::= = | <> | < | > | <= | >=

          <expression> ::= <term> | <adding-operator> <term>

            | <expression> <adding-operator> <term>

          <adding-operator> ::= + | -

          <term> ::= <factor> | <term> <multiplying-operator> <factor>

          <multiplying-operator> ::= \* | /

          <factor> ::= <ident> | <number> | ( <expression> )

**wenfa.txt**

S#A  
A#B  
B#CDEF  
C#aG;  
C#  
G#m=n  
G#G,m=n  
D#bJ;  
D#  
J#m  
J#J,m  
E#Ecm;B;  
E#  
F#m:=K  
F#dm  
F#eLf  
F#hMiF  
F#jMlF  
F#  
L#F  
L#L;F  
M#kK  
M#KNK  
N#=  
N#<>  
N#<  
N#>  
N#<=  
N#>=  
K#O  
K#PO  
K#KPO  
P#+  
P#-  
O#Q  
O#ORQ  
R#\*  
R#/  
Q#m  
Q#n  
Q#(K)

**first.txt**

A abcmdehj%  
B abcmdehj%  
C a%  
D b%  
E c%  
F mdehj%  
G m  
J m  
K +-  
L mdehj;%  
M +-k  
N =<>  
O mn(  
P +-  
Q mn(  
R \*/