编译器专题实验报告

实验五:语义分析

实验内容:

目的: 构建语法制导的语义分析程序能在语法分析的同时生成符号表和中间语言代码，并输出结果到文件中。

功能:

- SLR(1)制导的语义分析框架实现;
- 中间语言代码形式，三元式或四元式，或逆波兰表达式

实验结果:

具体实验要求：（必做部分）

- 根据之前的代码实现的SLR分析表，设计语法制导翻译过程，设计中间代码四元式或者三元式分析过程
- 输入：s=a+b+c+(a*a)
- 输出：四元式（1） （* a a T1） （2） （+ a b T2） （3） （+ T2 c T3） ...

具体实验要求：（选做部分）

- 出错判断，当二元运算符缺少运算对象等问题能够报错
- 可以实现if语句、while语句等任意控制语句的语义分析,比如实现if的四元式

C程序1:

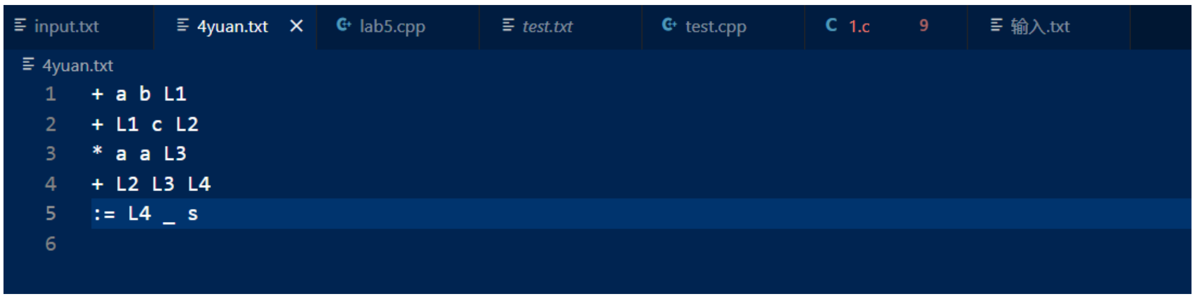
```
s=a+b+c+(a*a);
```

抽象语法树1:

```
IDENFR s
ASSIGNTK =
IDENFR a
PLUSTK +
IDENFR b
PLUSTK +
IDENFR c
PLUSTK +
LPARTK (
IDENFR a
MULTITK *
IDENFR a
RPARTK )
SEMICOLONTK ;
```

生成的四元式1:

```
+ a b L1
+ L1 c L2
* a a L3
+ L2 L3 L4
:= L4 _ s
```



```
+ a b L1
+ L1 c L2
* a a L3
+ L2 L3 L4
:= L4 _ s
```

C程序2：

```c
if(a){
    a=a+b;
}
else{
    a=c+d;
}
```

抽象语法树2：

```
IFTK if
LPARTK (
IDENFR a
RPARTK )
LBRATK {
IDENFR a
ASSIGNTK =
IDENFR a
PLUSTK +
IDENFR b
SEMICOLONTK ;
RBRATK }
ELSETK else
LBRATK {
IDENFR a
ASSIGNTK =
IDENFR c
PLUSTK +
IDENFR d
SEMICOLONTK ;
RBRATK }
```

文法：

```
17
S->A
A->P{A}
A->H
A->W{A}
W->wEo
C->fE
```

```
P->C{A}e
H->idE
H->E
E->E+T
E->E-T
E->T
T->T*F
T->T/F
T->F
F->(E)
F->i
```

上面的文法中 `f` 代表if； `e` 代表else；

生成的四元式2：

```
jnz a _ 0
+ a b L1
:= L1 _ a
j _ _ 0
+ c d L2
:= L2 _ a
```

另外遇到的问题和解决思路（可选）：

1. 首先通过第二次实验的内容，将C程序(输入串)通过词法分析为抽象语法树，再将抽象语法树作为语法分析部分的输入，产生四元式。
2. 同时结合第四次实验的内容生成语法分析的内容
3. 根据analyse函数中的规约顺序，向txt文件中输入四元式
4. 增加了一个symbol的字符串容器，用于存放规约后的临时变量

代码很原创（可选）：

```
#include <iostream>
#include <fstream>
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <cctype>
#include <vector>
#include <string>
#include <queue>
#include <map>
#include <set>
#include <sstream>
#define MAX 507
#define DEBUG
/*Author : byj*/
using namespace std;
vector<string> symbol; // 存放归约的符号
class WF
{
public:
    string left, right;
    int back;
```

```cpp
    int id;
    WF(char s1[], char s2[], int x, int y)
    {
        left = s1;
        right = s2;
        back = x;
        id = y;
    }
    WF(const string &s1, const string &s2, int x, int y)
    {
        left = s1;
        right = s2;
        back = x;
        id = y;
    }
    bool operator<(const WF &a) const
    {
        if (left == a.left)
            return right < a.right;
        return left < a.left;
    }
    bool operator==(const WF &a) const
    {
        return (left == a.left) && (right == a.right);
    }
    void print()
    {
        printf("%s->%s\n", left.c_str(), right.c_str());
    }
};

class Closure
{
public:
    vector<WF> element;
    void print(string str)
    {
        printf("%-15s%-15s\n", "", str.c_str());
        for (int i = 0; i < element.size(); i++)
            element[i].print();
    }
    bool operator==(const Closure &a) const
    {
        if (a.element.size() != element.size())
            return false;
        for (int i = 0; i < a.element.size(); i++)
            if (element[i] == a.element[i])
                continue;
            else
                return false;
        return true;
    }
};

struct Content
{
    int type;
    int num;
    string out;
    Content() { type = -1; }
    Content(int a, int b)
        : type(a), num(b) {}
};

vector<WF> wf;
map<string, vector<int>> dic;
map<string, vector<int>> VN_set;
map<string, bool> vis;
string start = "S";
vector<Closure> collection;
vector<WF> items;
char CH = '$';
int go[MAX][MAX];
int to[MAX];
vector<char> V;
bool used[MAX];
Content action[MAX][MAX];
int Goto[MAX][MAX];
map<string, set<char>> first;
map<string, set<char>> follow;

void make_item()
{
    memset(to, -1, sizeof(-1));
    for (int i = 0; i < wf.size(); i++)
        VN_set[wf[i].left].push_back(i);
    for (int i = 0; i < wf.size(); i++)
        for (int j = 0; j <= wf[i].right.length(); j++)
        {
            string temp = wf[i].right;
            temp.insert(temp.begin() + j, CH);
            dic[wf[i].left].push_back(items.size());
            if (j)
                to[items.size() - 1] = items.size();
            items.push_back(WF(wf[i].left, temp, i, items.size()));
        }
#ifdef DEBUG
    puts("-------------------------项目表-------------------------");
    for (int i = 0; i < items.size(); i++)
        printf("%s->%s back:%d id:%d\n", items[i].left.c_str(),
items[i].right.c_str(), items[i].back, items[i].id);
    puts("--------------------------------------------------------");
#endif
}

void dfs(const string &x)
{
    if (vis[x])
        return;
    vis[x] = 1;
    vector<int> &id = VN_set[x];
    for (int i = 0; i < id.size(); i++)
    {
        string &left = wf[id[i]].left;
        string &right = wf[id[i]].right;
        for (int j = 0; j < right.length(); j++)
            if (isupper(right[j]))
            {
                dfs(right.substr(j, 1));
                set<char> &temp = first[right.substr(j, 1)];
                set<char>::iterator it = temp.begin();
                bool flag = true;
                for (; it != temp.end(); it++)
                {
                    if (*it == '~')
                        flag = false;
                    first[left].insert(*it);
                }
                if (flag)
                    break;
            }
            else
            {
                first[left].insert(right[j]);
                break;
            }
    }
}

void make_first()
{
    vis.clear();
    map<string, vector<int>>::iterator it2 = dic.begin();
    for (; it2 != dic.end(); it2++)
        if (vis[it2->first])
            continue;
        else
            dfs(it2->first);
#ifdef DEBUG
    puts("****************FIRST集***************************");
    map<string, set<char>>::iterator it = first.begin();
    for (; it != first.end(); it++)
    {
        printf("FIRST(%s)={", it->first.c_str());
        set<char> &temp = it->second;
        set<char>::iterator it1 = temp.begin();
        bool flag = false;
        for (; it1 != temp.end(); it1++)
        {
            if (flag)
                printf(",");
            printf("%c", *it1);
            flag = true;
        }
        puts("}");
    }
#endif
}
```

```cpp

void append(const string &str1, const string &str2)
{
    set<char> &from = follow[str1];
    set<char> &to = follow[str2];
    set<char>::iterator it = from.begin();
    for (; it != from.end(); it++)
        to.insert(*it);
}

bool _check(const vector<int> &id, const string str)
{
    for (int i = 0; i < id.size(); i++)
    {
        int x = id[i];
        if (wf[x].right == str)
            return true;
    }
    return false;
}

void make_follow()
{
    while (true)
    {
        bool goon = false;
        map<string, vector<int>>::iterator it2 = VN_set.begin();
        for (; it2 != VN_set.end(); it2++)
        {
            vector<int> &id = it2->second;
            for (int i = 0; i < id.size(); i++)
            {
                bool flag = true;
                WF &tt = wf[id[i]];
                string &left = tt.left;
                const string &right = tt.right;
                for (int j = right.length() - 1; j >= 0; j--)
                    if (isupper(right[j]))
                    {
                        if (flag)
                        {
                            int tx = follow[right.substr(j, 1)].size();
                            append(left, right.substr(j, 1));
                            int tx1 = follow[right.substr(j, 1)].size();
                            if (tx1 > tx)
                                goon = true;
                            if (_check(id, "~"))
                                flag = false;
                        }
                        for (int k = j + 1; k < right.length(); k++)
                            if (isupper(right[k]))
                            {
                                string idd = right.substr(k, 1);
                                set<char> &from = first[idd];
                                set<char> &to = follow[right.substr(j, 1)];
                                set<char>::iterator it1 = from.begin();
                                int tx = follow[right.substr(j, 1)].size();
                                for (; it1 != from.end(); it1++)
                                    if (*it1 != '~')
                                        to.insert(*it1);
                                int tx1 = follow[right.substr(j, 1)].size();
                                if (tx1 > tx)
                                    goon = true;
                                if (_check(id, "~"))
                                    break;
                            }
                            else
                            {
                                int tx = follow[right.substr(j, 1)].size();
                                follow[right.substr(j, 1)].insert(right[k]);
                                int tx1 = follow[right.substr(j, 1)].size();
                                if (tx1 > tx)
                                    goon = true;
                                break;
                            }
                    }
                    else
                        flag = false;
            }
        }
        if (!goon)
            break;
    }
#ifdef DEBUG
    puts("***************FOLLOW集*******************");
    map<string, set<char>>::iterator it = follow.begin();
    for (; it != follow.end(); it++)
    {
        printf("FOLLOW(%s)={", it->first.c_str());
        set<char> &temp = it->second;
        // if ( it->first[0] == 'S' )
        temp.insert('#');
        set<char>::iterator it1 = temp.begin();
        bool flag = false;
        for (; it1 != temp.end(); it1++)
        {
            if (flag)
                printf(",");
            printf("%c", *it1);
            flag = true;
        }
        puts("}");
    }
#endif
}

void make_set()
{
    bool has[MAX];
    for (int i = 0; i < items.size(); i++)
        if (items[i].left[0] == 'S' && items[i].right[0] == CH)
        {
            Closure temp;
            string &str = items[i].right;
            vector<WF> &element = temp.element;
            element.push_back(items[i]);
            int x = 0;
            for (x = 0; x < str.length(); x++)
                if (str[x] == CH)
                    break;
            /*if ( x != str.length()-1 )
            {
                string tt = str.substr(x+1,1);
                vector<int>& id = dic[tt];
                for ( int j = 0 ; j < id.size() ; j++ )
                {
                    int tx = id[j];
                    //items[tx].print();
                    if ( items[tx].right[0] == CH )
                        element.push_back ( items[tx] );
                }
            }*/
            memset(has, 0, sizeof(has));
            has[i] = 1;
            if (x != str.length() - 1)
            {
                queue<string> q;
                q.push(str.substr(x + 1, 1));
                while (!q.empty())
                {
                    string u = q.front();
                    q.pop();
                    vector<int> &id = dic[u];
                    for (int j = 0; j < id.size(); j++)
                    {
                        int tx = id[j];
                        if (items[tx].right[0] == CH)
                        {
                            if (has[tx])
                                continue;
                            has[tx] = 1;
                            if (isupper(items[tx].right[1]))
                                q.push(items[tx].right.substr(1, 1));
                            element.push_back(items[tx]);
                        }
                    }
                }
            }
            collection.push_back(temp);
        }
    for (int i = 0; i < collection.size(); i++)
    {
        map<int, Closure> temp;
        for (int j = 0; j < collection[i].element.size(); j++)
        {
            string str = collection[i].element[j].right;
            int x = 0;
            for (; x < str.length(); x++)
```

```
                if (str[x] == CH)
                    break;
            if (x == str.length() - 1)
                continue;
            int y = str[x + 1];
            int ii;
            // cout << i << "previous: " << str << endl;
            str.erase(str.begin() + x);
            str.insert(str.begin() + x + 1, CH);
            // cout << i <<"after: " << str << endl;
            WF cmp = WF(collection[i].element[j].left, str, -1, -1);
            for (int k = 0; k < items.size(); k++)
                if (items[k] == cmp)
                {
                    ii = k;
                    break;
                }
            // string& str1 = items[ii].right;
            memset(has, 0, sizeof(has));
            vector<WF> &element = temp[y].element;
            element.push_back(items[ii]);
            has[ii] = 1;
            x++;
            /*if ( x != str.length()-1 )
            {
                string tt = str.substr(x+1,1);
                vector<int>& id = dic[tt];
                for ( int j = 0 ; j < id.size() ; j++ )
                {
                   int tx = id[j];
                   //items[tx].print();
                   if ( items[tx].right[0] == CH )
                       element.push_back ( items[tx] );
                }
            }*/
            if (x != str.length() - 1)
            {
                queue<string> q;
                q.push(str.substr(x + 1, 1));
                while (!q.empty())
                {
                    string u = q.front();
                    q.pop();
                    vector<int> &id = dic[u];
                    for (int j = 0; j < id.size(); j++)
                    {
                        int tx = id[j];
                        if (items[tx].right[0] == CH)
                        {
                            if (has[tx])
                                continue;
                            has[tx] = 1;
                            if (isupper(items[tx].right[1]))
                                q.push(items[tx].right.substr(1, 1));
                            element.push_back(items[tx]);
                        }
```

```cpp
                        }
                    }
                }
            }
            map<int, Closure>::iterator it = temp.begin();
            for (; it != temp.end(); it++)
                collection.push_back(it->second);
            for (int i = 0; i < collection.size(); i++)
                sort(collection[i].element.begin(), collection[i].element.end());
            for (int i = 0; i < collection.size(); i++)
                for (int j = i + 1; j < collection.size(); j++)
                    if (collection[i] == collection[j])
                        collection.erase(collection.begin() + j);
        }
#ifdef DEBUG
    puts("-------------CLOSURE---------------------");
    stringstream sin;
    for (int i = 0; i < collection.size(); i++)
    {
        sin.clear();
        string out;
        sin << "closure-I" << i;
        sin >> out;
        collection[i].print(out);
    }
    puts("");
#endif
}

void make_V()
{
    memset(used, 0, sizeof(used));
    for (int i = 0; i < wf.size(); i++)
    {
        string &str = wf[i].left;
        for (int j = 0; j < str.length(); j++)
        {
            if (used[str[j]])
                continue;
            used[str[j]] = 1;
            V.push_back(str[j]);
        }
        string &str1 = wf[i].right;
        for (int j = 0; j < str1.length(); j++)
        {
            if (used[str1[j]])
                continue;
            used[str1[j]] = 1;
            V.push_back(str1[j]);
        }
    }
    sort(V.begin(), V.end());
    V.push_back('#');
}

void make_cmp(vector<WF> &cmp1, int i, char ch)
```

```cpp
{
    for (int j = 0; j < collection[i].element.size(); j++)
    {
        string str = collection[i].element[j].right;
        int k;
        for (k = 0; k < str.length(); k++)
            if (str[k] == CH)
                break;
        if (k != str.length() - 1 && str[k + 1] == ch)
        {
            str.erase(str.begin() + k);
            str.insert(str.begin() + k + 1, CH);
            cmp1.push_back(WF(collection[i].element[j].left, str, -1, -1));
        }
    }
    sort(cmp1.begin(), cmp1.end());
}

void make_go()
{
    memset(go, -1, sizeof(go));
    int m = collection.size();
    /*for ( int i = 0 ; i < m ; i++ )
        for ( int j = 0 ; j < collection[i].element.size() ; j++ )
        {
            string left = collection[i].element[j].left;
            string str = collection[i].element[j].right;
            int x = 0;
            for ( ; x < str.length() ; x++ )
               if ( str[x] == CH ) break;
            if ( x == str.length()-1 )
                continue;
            int y = str[x+1];
           //cout << "before : " << str << endl;
            str.erase ( str.begin()+x);
            str.insert ( str.begin()+x+1 , CH );
           //cout << "after : " << str << endl;
            WF cmp = WF ( collection[i].element[j].left , str , -1 , -1 );
            for ( int k = 0 ; k < m ; k++ )
            {
                bool flag = false;
                for ( int t = 0 ; t < collection[k].element.size() ; t++ )
                {
                    if ( cmp == collection[k].element[t] )
                    {
                        flag = true;
                        break;
                    }
                }
                if ( flag )
                {
                    go[i][y] = k;
                }
            }
        }*/
    for (int t = 0; t < V.size(); t++)
    {
        char ch = V[t];
        for (int i = 0; i < m; i++)
        {
            vector<WF> cmp1;
            make_cmp(cmp1, i, ch);
            cout << cmp1.size() << endl;
            if (cmp1.size() == 0)
                continue;
            for (int j = 0; j < m; j++)
            {
                vector<WF> cmp2;
                for (int k = 0; k < collection[j].element.size(); k++)
                {
                    string &str = collection[j].element[k].right;
                    int x;
                    for (x = 0; x < str.length(); x++)
                        if (str[x] == CH)
                            break;
                    if (x && str[x - 1] == ch)
                        cmp2.push_back(WF(collection[j].element[k].left, str,
-1, -1));
                }
                sort(cmp2.begin(), cmp2.end());
                cout << cmp2.size() << endl;
                bool flag = true;
                if (cmp2.size() != cmp1.size())
                    continue;
                cout << cmp1.size() << endl;
                for (int k = 0; k < cmp1.size(); k++)
                    if (cmp1[k] == cmp2[k])
                        continue;
                    else
                        flag = false;
                cout << "out " << endl;
                if (flag)
                    go[i][ch] = j;
            }
            // cout << "YES" << endl;
        }
    }
#ifdef DEBUG
    puts("---------------EDGE----------------------");
    stringstream sin;
    string out;
    for (int i = 0; i < m; i++)
        for (int j = 0; j < m; j++)
            for (int k = 0; k < MAX; k++)
                if (go[i][k] == j)
                {
                    sin.clear();
                    sin << "I" << i << "--" << (char)(k) << "--I" << j;
                    sin >> out;
                    printf("%s\n", out.c_str());
                }
#endif
```

```
}
bool cmp(char a, char b)
{
    if (a >= 'A' && a <= 'Z' && b >= 'a' && b <= 'z')
        return 0;
    if (a >= 'a' && a <= 'z' && b >= 'A' && b <= 'Z')
        return 1;
    return a < b;
}
void make_table()
{
    memset(Goto, -1, sizeof(Goto));
    int m = collection.size();
    for (int i = 0; i < m; i++)
        for (char ch : V) // 优化
        {
            int x = go[i][ch];
            if (x == -1)
                continue;
            if (!isupper(ch))
                action[i][ch] = Content(0, x);
            else
                Goto[i][ch] = x;
        }

    // 规约
    for (int i = 0; i < m; i++)
        for (int j = 0; j < collection[i].element.size(); j++)
        {
            WF &tt = collection[i].element[j];
            if (tt.right[tt.right.length() - 1] == CH)
            {
                if (tt.left[0] == 'S')
                    action[i]['#'] = Content(2, -1);
                else
                    for (int k = 0; k < V.size(); k++)
                    {
                        int y = V[k];
                        if (!follow[tt.left].count(V[k]))
                            continue;
                        // 判断是否是slr(1)文法,如果在相同位置已经填过数据，则发生了冲突，不是slr(1)文法
                        if (action[i][y].type != -1)
                        {
                            cout << "状态" << i << " " << "发生冲突的符号" << (char)y << endl;
                            cout << "不是slr(1)文法" << endl;
                        }

                        action[i][y] = Content(1, tt.back);
                    }
            }
        }
#ifdef DEBUG
    puts("------------------------------------------SLR(1)分析表------------------------------------------------------");
    sort(V.begin(), V.end(), cmp);
    printf("%10s%5c%5s", "|", V[0], "|");
    for (int i = 1; i < V.size(); i++)
        printf("%5c%5s", V[i], "|");
    puts("");
    for (int i = 0; i < (V.size() + 1) * 10; i++)
        printf("-");
    puts("");
    stringstream sin;
    for (int i = 0; i < collection.size(); i++)
    {
        printf("%5d%5s", i, "|");
        for (int j = 0; j < V.size(); j++)
        {
            char ch = V[j];
            if (isupper(ch))
            {
                if (Goto[i][ch] == -1)
                    printf("%10s", "|");
                else
                    printf("%5d%5s", Goto[i][ch], "|");
            }
            else
            {
                sin.clear();
                if (action[i][ch].type == -1)
                    printf("%10s", "|");
                else
                {
                    Content &temp = action[i][ch];
                    if (temp.type == 0)
                        sin << "S";
                    if (temp.type == 1)
                        sin << "R";
                    if (temp.type == 2)
                        sin << "acc";
                    if (temp.num != -1)
                        sin << temp.num;
                    sin >> temp.out;
                    printf("%7s%3s", temp.out.c_str(), "|");
                }
            }
        }
        puts("");
    }
    for (int i = 0; i < (V.size() + 1) * 10; i++)
        printf("-");
    puts("");
#endif
}

void print(string s1, string s2, string s3, string s4, string s5, string s6,
string s7)
{
    printf("%-15s|%-15s%-15s%-20s|%-15s%-15s%-15s\n", s1.c_str(), s2.c_str(),
s3.c_str(), s4.c_str(), s5.c_str(),
```

```cpp
            s6.c_str(), s7.c_str());
}

string get_steps(int x)
{
    stringstream sin;
    sin << x;
    string ret;
    sin >> ret;
    return ret;
}

template <class T>
string get_stk(vector<T> stk)
{
    stringstream sin;
    for (int i = 0; i < stk.size(); i++)
        sin << stk[i];
    string ret;
    sin >> ret;
    return ret;
}

string get_shift(WF &temp)
{
    stringstream sin;
    sin << "reduce(" << temp.left << "->" << temp.right << ")";
    string out;
    sin >> out;
    return out;
}

void analyse(string src)
{
    std::ofstream outfile("4yuan.txt");
    print("steps", "op-stack", "input", "operation", "state-stack", "ACTION",
"GOTO");
    vector<char> op_stack;
    vector<int> st_stack;
    vector<string> symbol2;
    string label;
    auto it = symbol.begin();
    int n = 1;
    src += "#";
    op_stack.push_back('#');
    st_stack.push_back(0);
    int steps = 1;
    for (int i = 0; i < src.length(); i++)
    {
        char u = src[i];
        int top = st_stack[st_stack.size() - 1];
        // action两个参数：状态集合编号，输入符号
        Content &act = action[top][u];
        // cout << "YES : " << i << " " << u << " " << top << " " << act.type
<< endl;
        if (act.type == 0)
        {
            print(get_steps(steps++), get_stk(op_stack), src.substr(i),
"shift", get_stk(st_stack), act.out, "");
            op_stack.push_back(u);
            st_stack.push_back(act.num);
            if (u == 'i')
            {
                symbol2.push_back(*it);
                ++it;
            }
        }
        else if (act.type == 1)
        {
            // act.num是当前规约的产生式编号
            string s1;
            string s2;
            switch (act.num)
            {
            case 9:
                // cout << "E->E+T" << endl;
                // cout << "symbol2.size:" << symbol2.size() << endl;
                s1 = symbol2.back();
                symbol2.pop_back();
                s2 = symbol2.back();
                symbol2.pop_back();
                label = "L" + std::to_string(n);
                n++;
                symbol2.push_back(label);
                outfile << "+ " << s2 << ' ' << s1 << ' ' << label << endl;
                break;
            case 10:
                // cout << "E->E-T" << endl;
                // cout << "symbol2.size:" << symbol2.size() << endl;
                s1 = symbol2.back();
                symbol2.pop_back();
                s2 = symbol2.back();
                symbol2.pop_back();
                label = "L" + std::to_string(n);
                n++;
                symbol2.push_back(label);
                outfile << "- " << s2 << ' ' << s1 << ' ' << label << endl;
                break;
                break;
            case 12:
                // cout << "T->T*F" << endl;
                // cout << "symbol2.size:" << symbol2.size() << endl;
                s1 = symbol2.back();
                symbol2.pop_back();
                s2 = symbol2.back();
                symbol2.pop_back();
                label = "L" + std::to_string(n);
                n++;
                symbol2.push_back(label);
                outfile << "* " << s2 << ' ' << s1 << ' ' << label << endl;
                break;
            case 13:
```

```cpp
                // cout << "T->E/F" << endl;
                // cout << "symbol2.size:" << symbol2.size() << endl;
                s1 = symbol2.back();
                symbol2.pop_back();
                s2 = symbol2.back();
                symbol2.pop_back();
                label = "L" + std::to_string(n);
                n++;
                symbol2.push_back(label);
                outfile << "/ " << s2 << ' ' << s1 << ' ' << label << endl;
                break;
            case 7: // 赋值
                // cout << "E=ID" << endl;
                // cout << "symbol2.size:" << symbol2.size() << endl;
                s1 = symbol2.back();
                symbol2.pop_back();
                s2 = symbol2.back();
                symbol2.pop_back();
                outfile << ":= " << s1 << ' ' << "_ " << s2 << endl;
                break;
            case 5: // if
                // cout << "if" << endl;
                // cout << "symbol2.size:" << symbol2.size() << endl;
                s1 = symbol2.back();
                outfile << "jnz " << s1 << " _ 0" << endl;
                break;
            case 6: // else
                // cout << "else" << endl;
                // cout << "symbol2.size:" << symbol2.size() << endl;
                outfile << "j _ _ 0" << endl;
                break;
            case 4: // while
                // cout << "while" << endl;
                // cout << "symbol2.size:" << symbol2.size() << endl;
                s1 = symbol2.back();
                outfile << "jnz " << s1 << " _ 0" << endl;
                break;
            case 3: // do
                outfile << "j _ _ 0" << endl;
                break;
            default:
                break;
            }
            WF &tt = wf[act.num];
            int y = st_stack[st_stack.size() - tt.right.length() - 1];
            int x = Goto[y][tt.left[0]];
            // cout << y << " " << tt.left[0] << " " << x << endl;
            print(get_steps(steps++), get_stk(op_stack), src.substr(i), get_shift(tt), get_stk(st_stack), act.out, get_steps(x));
            for (int j = 0; j < tt.right.length(); j++)
            {
                st_stack.pop_back();
                op_stack.pop_back();
            }
            op_stack.push_back(tt.left[0]);
            st_stack.push_back(x);
            i--;
        }
        else if (act.type == 2)
        {
            print(get_steps(steps++), get_stk(op_stack), src.substr(i), "Accept", get_stk(st_stack), act.out, "");
            // i--;
        }
        else
            continue;
    }
}
string toSymbol()
{
    std::ifstream infile("input.txt");
    std::string line = "";
    std::string word = "";

    if (infile.is_open())
    {
        while (infile >> word)
        {
            cout << word << endl;
            if (word == "IFTK")
            {
                line += "f";
                infile >> word;
                continue;
            }
            else if (word == "THENTK")
            {
                line += "t";
                infile >> word;
                continue;
            }
            else if (word == "ELSETK")
            {
                line += "e";
                infile >> word;
                continue;
            }
            else if (word == "IDENFR" || word == "INTCON")
            {
                line += 'i';
                infile >> word;
                symbol.push_back(word);
                continue;
            }
            else if (word == "PLUSTK")
            {
                line += "+";
                infile >> word;
                continue;
            }
            else if (word == "MULTITK")
            {
                line += "*";
                infile >> word;
                continue;
            }
            else if (word == "DIVTK")
            {
                line += "/";
                infile >> word;
                continue;
            }
            else if (word == "MINUTETK")
            {
                line += "-";
                infile >> word;
                continue;
            }
            else if (word == "LPARTK")
            {
                line += "(";
                infile >> word;
                continue;
            }
            else if (word == "RPARTK")
            {
                line += ")";
                infile >> word;
                continue;
            }
            else if (word == "ASSIGNTK")
            {
                line += "d";
                infile >> word;
                continue;
            }
            else if (word == "WHILETK")
            {
                line += "w";
                infile >> word;
                continue;
            }
            else if (word == "DOTK")
            {
                line += "o";
                infile >> word;
                continue;
            }
            else if (word == "LBRATK")
            {
                line += "{";
                infile >> word;
                continue;
            }
            else if (word == "RBRATK")
            {
                line += "}";
                infile >> word;
```

```cpp
                    continue;
                }
                else
                {
                    infile >> word;
                    continue;
                }
            }
            infile.close();
        }
        else
        {
            std::cout << "Unable to open file";
        }
        cout << "symbol.size:" << symbol.size() << endl;
        return line;
}
int main()
{
    int n;
    char s[MAX];
    while (~scanf("%d", &n))
    {
        for (int i = 0; i < n; i++)
        {
            scanf("%s", s);
            int len = strlen(s), j;
            for (j = 0; j < len; j++)
                if (s[j] == '-')
                    break;
            s[j] = 0;
            wf.push_back(WF(s, s + j + 2, -1, -1));
#ifdef DEBUG
            wf[wf.size() - 1].print();
#endif
        }
        make_item();
        make_first();
        make_follow();
        make_set();
        make_V();
        make_go();
        make_table();
        string str = toSymbol();
        cout << str << endl;
        analyse(str);
    }
}
```