

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) $(+ T2 c T3) \dots$

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:

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

抽象语法树1:

```
1  IDENFR s
2  ASSIGNTK =
3  IDENFR a
4  PLUSTK +
5  IDENFR b
6  PLUSTK +
7  IDENFR c
8  PLUSTK +
9  LPARTK (
10 IDENFR a
11 MULTITK *
12 IDENFR a
13 RPARTK )
14 SEMICOLONTK ;
```

生成的四元式1:

```
1   + a b L1
2   + L1 c L2
3   * a a L3
4   + L2 L3 L4
5   := L4 _ s
```

```
input.txt 4yuan.txt x lab5.cpp test.txt test.cpp 1.c 9 输入.txt
4yuan.txt
1   + a b L1
2   + L1 c L2
3   * a a L3
4   + L2 L3 L4
5   := L4 _ s
6
```

C程序2:

```
1   if(a){
2       a=a+b;
3   }
4   else{
5       a=c+d;
6   }
```

抽象语法树2:

```
1   IFTK if
2   LPARTK (
3   IDENFR a
4   RPARTK )
5   LBRATK {
6   IDENFR a
7   ASSIGNTK =
8   IDENFR a
9   PLUSTK +
10  IDENFR b
11  SEMICOLONTK ;
12  RBRATK }
13  ELSETK else
14  LBRATK {
15  IDENFR a
16  ASSIGNTK =
17  IDENFR c
18  PLUSTK +
19  IDENFR d
20  SEMICOLONTK ;
21  RBRATK }
```

文法:

```
1   17
2   S->A
3   A->P{A}
4   A->H
5   A->W{A}
6   W->wEo
7   C->fE
```

```

8   P->C{A}e
9   H->idE
10  H->E
11  E->E+T
12  E->E-T
13  E->T
14  T->T*F
15  T->T/F
16  T->F
17  F->(E)
18  F->i

```

上面的文法中 **f** 代表if; **e** 代表else;

生成的四元式2:

```

1   jnz a _ 0
2   + a b L1
3   := L1 _ a
4   j _ _ 0
5   + c d L2
6   := L2 _ a

```

另外遇到的问题 and 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) :

```

1   #include <iostream>
2   #include <fstream>
3   #include <cstdio>
4   #include <algorithm>
5   #include <cstring>
6   #include <cctype>
7   #include <vector>
8   #include <string>
9   #include <queue>
10  #include <map>
11  #include <set>
12  #include <sstream>
13  #define MAX 507
14  #define DEBUG
15  /*Author : byj*/
16  using namespace std;
17  vector<string> symbol; // 存放归约的符号
18  class WF
19  {
20  public:
21      string left, right;
22      int back;

```

```

23     int id;
24     WF(char s1[], char s2[], int x, int y)
25     {
26         left = s1;
27         right = s2;
28         back = x;
29         id = y;
30     }
31     WF(const string &s1, const string &s2, int x, int y)
32     {
33         left = s1;
34         right = s2;
35         back = x;
36         id = y;
37     }
38     bool operator<(const WF &a) const
39     {
40         if (left == a.left)
41             return right < a.right;
42         return left < a.left;
43     }
44     bool operator==(const WF &a) const
45     {
46         return (left == a.left) && (right == a.right);
47     }
48     void print()
49     {
50         printf("%s->%s\n", left.c_str(), right.c_str());
51     }
52 };
53
54 class Closure
55 {
56 public:
57     vector<WF> element;
58     void print(string str)
59     {
60         printf("%-15s%-15s\n", "", str.c_str());
61         for (int i = 0; i < element.size(); i++)
62             element[i].print();
63     }
64     bool operator==(const Closure &a) const
65     {
66         if (a.element.size() != element.size())
67             return false;
68         for (int i = 0; i < a.element.size(); i++)
69             if (element[i] == a.element[i])
70                 continue;
71             else
72                 return false;
73         return true;
74     }
75 };
76
77 struct Content
78 {

```

```

79     int type;
80     int num;
81     string out;
82     Content() { type = -1; }
83     Content(int a, int b)
84         : type(a), num(b) {}
85 };
86
87 vector<WF> wf;
88 map<string, vector<int>> dic;
89 map<string, vector<int>> VN_set;
90 map<string, bool> vis;
91 string start = "S";
92 vector<Closure> collection;
93 vector<WF> items;
94 char CH = '$';
95 int go[MAX][MAX];
96 int to[MAX];
97 vector<char> V;
98 bool used[MAX];
99 Content action[MAX][MAX];
100 int Goto[MAX][MAX];
101 map<string, set<char>> first;
102 map<string, set<char>> follow;
103
104 void make_item()
105 {
106     memset(to, -1, sizeof(-1));
107     for (int i = 0; i < wf.size(); i++)
108         VN_set[wf[i].left].push_back(i);
109     for (int i = 0; i < wf.size(); i++)
110         for (int j = 0; j <= wf[i].right.length(); j++)
111         {
112             string temp = wf[i].right;
113             temp.insert(temp.begin() + j, CH);
114             dic[wf[i].left].push_back(items.size());
115             if (j)
116                 to[items.size() - 1] = items.size();
117             items.push_back(WF(wf[i].left, temp, i, items.size()));
118         }
119 #ifdef DEBUG
120     puts("-----项目表-----");
121     for (int i = 0; i < items.size(); i++)
122         printf("%s->%s back:%d id:%d\n", items[i].left.c_str(),
123             items[i].right.c_str(), items[i].back, items[i].id);
124     puts("-----");
125 #endif
126 }
127
128 void dfs(const string &x)
129 {
130     if (vis[x])
131         return;
132     vis[x] = 1;
133     vector<int> &id = VN_set[x];
134     for (int i = 0; i < id.size(); i++)

```

```

134     {
135         string &left = wf[id[i]].left;
136         string &right = wf[id[i]].right;
137         for (int j = 0; j < right.length(); j++)
138             if (isupper(right[j]))
139                 {
140                     dfs(right.substr(j, 1));
141                     set<char> &temp = first[right.substr(j, 1)];
142                     set<char>::iterator it = temp.begin();
143                     bool flag = true;
144                     for (; it != temp.end(); it++)
145                         {
146                             if (*it == '~')
147                                 flag = false;
148                             first[left].insert(*it);
149                         }
150                     if (flag)
151                         break;
152                 }
153             else
154                 {
155                     first[left].insert(right[j]);
156                     break;
157                 }
158     }
159 }
160
161 void make_first()
162 {
163     vis.clear();
164     map<string, vector<int>>::iterator it2 = dic.begin();
165     for (; it2 != dic.end(); it2++)
166         if (vis[it2->first])
167             continue;
168         else
169             dfs(it2->first);
170 #ifdef DEBUG
171     puts("*****FIRST集*****");
172     map<string, set<char>>::iterator it = first.begin();
173     for (; it != first.end(); it++)
174         {
175             printf("FIRST(%s)={", it->first.c_str());
176             set<char> &temp = it->second;
177             set<char>::iterator it1 = temp.begin();
178             bool flag = false;
179             for (; it1 != temp.end(); it1++)
180                 {
181                     if (flag)
182                         printf(", ");
183                     printf("%c", *it1);
184                     flag = true;
185                 }
186             puts("}");
187         }
188 #endif
189 }

```

```

190
191 void append(const string &str1, const string &str2)
192 {
193     set<char> &from = follow[str1];
194     set<char> &to = follow[str2];
195     set<char>::iterator it = from.begin();
196     for (; it != from.end(); it++)
197         to.insert(*it);
198 }
199
200 bool _check(const vector<int> &id, const string str)
201 {
202     for (int i = 0; i < id.size(); i++)
203     {
204         int x = id[i];
205         if (wf[x].right == str)
206             return true;
207     }
208     return false;
209 }
210
211 void make_follow()
212 {
213     while (true)
214     {
215         bool goon = false;
216         map<string, vector<int>>::iterator it2 = VN_set.begin();
217         for (; it2 != VN_set.end(); it2++)
218         {
219             vector<int> &id = it2->second;
220             for (int i = 0; i < id.size(); i++)
221             {
222                 bool flag = true;
223                 WF &tt = wf[id[i]];
224                 string &left = tt.left;
225                 const string &right = tt.right;
226                 for (int j = right.length() - 1; j >= 0; j--)
227                     if (isupper(right[j]))
228                     {
229                         if (flag)
230                         {
231                             int tx = follow[right.substr(j, 1)].size();
232                             append(left, right.substr(j, 1));
233                             int tx1 = follow[right.substr(j, 1)].size();
234                             if (tx1 > tx)
235                                 goon = true;
236                             if (_check(id, "~"))
237                                 flag = false;
238                         }
239                         for (int k = j + 1; k < right.length(); k++)
240                             if (isupper(right[k]))
241                             {
242                                 string idd = right.substr(k, 1);
243                                 set<char> &from = first[idd];
244                                 set<char> &to = follow[right.substr(j, 1)];
245                                 set<char>::iterator it1 = from.begin();

```

```

246         int tx = follow[right.substr(j, 1)].size();
247         for (; it1 != from.end(); it1++)
248             if (*it1 != '~')
249                 to.insert(*it1);
250         int tx1 = follow[right.substr(j, 1)].size();
251         if (tx1 > tx)
252             goon = true;
253         if (_check(id, "~"))
254             break;
255     }
256     else
257     {
258         int tx = follow[right.substr(j, 1)].size();
259         follow[right.substr(j, 1)].insert(right[k]);
260         int tx1 = follow[right.substr(j, 1)].size();
261         if (tx1 > tx)
262             goon = true;
263         break;
264     }
265 }
266 else
267     flag = false;
268 }
269 }
270 if (!goon)
271     break;
272 }
273 #ifdef DEBUG
274     puts("*****FOLLOW集*****");
275     map<string, set<char>>::iterator it = follow.begin();
276     for (; it != follow.end(); it++)
277     {
278         printf("FOLLOW(%s)={", it->first.c_str());
279         set<char> &temp = it->second;
280         // if ( it->first[0] == 'S' )
281         temp.insert('#');
282         set<char>::iterator it1 = temp.begin();
283         bool flag = false;
284         for (; it1 != temp.end(); it1++)
285         {
286             if (flag)
287                 printf(",");
288             printf("%c", *it1);
289             flag = true;
290         }
291         puts("}");
292     }
293 #endif
294 }
295
296 void make_set()
297 {
298     bool has[MAX];
299     for (int i = 0; i < items.size(); i++)
300         if (items[i].left[0] == 'S' && items[i].right[0] == CH)
301             {

```



```

302         Closure temp;
303         string &str = items[i].right;
304         vector<WF> &element = temp.element;
305         element.push_back(items[i]);
306         int x = 0;
307         for (x = 0; x < str.length(); x++)
308             if (str[x] == CH)
309                 break;
310         /*if ( x != str.length()-1 )
311         {
312             string tt = str.substr(x+1,1);
313             vector<int>& id = dic[tt];
314             for ( int j = 0 ; j < id.size() ; j++ )
315             {
316                 int tx = id[j];
317                 //items[tx].print();
318                 if ( items[tx].right[0] == CH )
319                     element.push_back ( items[tx] );
320             }
321         }*/
322         memset(has, 0, sizeof(has));
323         has[i] = 1;
324         if (x != str.length() - 1)
325         {
326             queue<string> q;
327             q.push(str.substr(x + 1, 1));
328             while (!q.empty())
329             {
330                 string u = q.front();
331                 q.pop();
332                 vector<int> &id = dic[u];
333                 for (int j = 0; j < id.size(); j++)
334                 {
335                     int tx = id[j];
336                     if (items[tx].right[0] == CH)
337                     {
338                         if (has[tx])
339                             continue;
340                         has[tx] = 1;
341                         if (isupper(items[tx].right[1]))
342                             q.push(items[tx].right.substr(1, 1));
343                         element.push_back(items[tx]);
344                     }
345                 }
346             }
347         }
348         collection.push_back(temp);
349     }
350     for (int i = 0; i < collection.size(); i++)
351     {
352         map<int, Closure> temp;
353         for (int j = 0; j < collection[i].element.size(); j++)
354         {
355             string str = collection[i].element[j].right;
356             int x = 0;
357             for (; x < str.length(); x++)

```

```

358         if (str[x] == CH)
359             break;
360     if (x == str.length() - 1)
361         continue;
362     int y = str[x + 1];
363     int ii;
364     // cout << i << "previous: " << str << endl;
365     str.erase(str.begin() + x);
366     str.insert(str.begin() + x + 1, CH);
367     // cout << i << "after: " << str << endl;
368     WF cmp = WF(collection[i].element[j].left, str, -1, -1);
369     for (int k = 0; k < items.size(); k++)
370         if (items[k] == cmp)
371         {
372             ii = k;
373             break;
374         }
375     // string& str1 = items[ii].right;
376     memset(has, 0, sizeof(has));
377     vector<WF> &element = temp[y].element;
378     element.push_back(items[ii]);
379     has[ii] = 1;
380     x++;
381     /*if ( x != str.length()-1 )
382     {
383         string tt = str.substr(x+1,1);
384         vector<int>& id = dic[tt];
385         for ( int j = 0 ; j < id.size() ; j++ )
386         {
387             int tx = id[j];
388             //items[tx].print();
389             if ( items[tx].right[0] == CH )
390                 element.push_back ( items[tx] );
391         }
392     }*/
393     if (x != str.length() - 1)
394     {
395         queue<string> q;
396         q.push(str.substr(x + 1, 1));
397         while (!q.empty())
398         {
399             string u = q.front();
400             q.pop();
401             vector<int> &id = dic[u];
402             for (int j = 0; j < id.size(); j++)
403             {
404                 int tx = id[j];
405                 if (items[tx].right[0] == CH)
406                 {
407                     if (has[tx])
408                         continue;
409                     has[tx] = 1;
410                     if (isupper(items[tx].right[1]))
411                         q.push(items[tx].right.substr(1, 1));
412                     element.push_back(items[tx]);
413                 }

```

```

414         }
415     }
416 }
417 }
418 map<int, Closure>::iterator it = temp.begin();
419 for (; it != temp.end(); it++)
420     collection.push_back(it->second);
421 for (int i = 0; i < collection.size(); i++)
422     sort(collection[i].element.begin(), collection[i].element.end());
423 for (int i = 0; i < collection.size(); i++)
424     for (int j = i + 1; j < collection.size(); j++)
425         if (collection[i] == collection[j])
426             collection.erase(collection.begin() + j);
427 }
428 #ifdef DEBUG
429     puts("-----CLOSURE-----");
430     stringstream sin;
431     for (int i = 0; i < collection.size(); i++)
432     {
433         sin.clear();
434         string out;
435         sin << "closure-I" << i;
436         sin >> out;
437         collection[i].print(out);
438     }
439     puts("");
440 #endif
441 }
442
443 void make_V()
444 {
445     memset(used, 0, sizeof(used));
446     for (int i = 0; i < wf.size(); i++)
447     {
448         string &str = wf[i].left;
449         for (int j = 0; j < str.length(); j++)
450         {
451             if (used[str[j]])
452                 continue;
453             used[str[j]] = 1;
454             V.push_back(str[j]);
455         }
456         string &str1 = wf[i].right;
457         for (int j = 0; j < str1.length(); j++)
458         {
459             if (used[str1[j]])
460                 continue;
461             used[str1[j]] = 1;
462             V.push_back(str1[j]);
463         }
464     }
465     sort(V.begin(), V.end());
466     V.push_back('#');
467 }
468
469 void make_cmp(vector<WF> &cmp1, int i, char ch)

```

```

470 {
471     for (int j = 0; j < collection[i].element.size(); j++)
472     {
473         string str = collection[i].element[j].right;
474         int k;
475         for (k = 0; k < str.length(); k++)
476             if (str[k] == CH)
477                 break;
478         if (k != str.length() - 1 && str[k + 1] == ch)
479         {
480             str.erase(str.begin() + k);
481             str.insert(str.begin() + k + 1, CH);
482             cmp1.push_back(WF(collection[i].element[j].left, str, -1, -1));
483         }
484     }
485     sort(cmp1.begin(), cmp1.end());
486 }
487
488 void make_go()
489 {
490     memset(go, -1, sizeof(go));
491     int m = collection.size();
492     /*for ( int i = 0 ; i < m ; i++ )
493         for ( int j = 0 ; j < collection[i].element.size() ; j++ )
494         {
495             string left = collection[i].element[j].left;
496             string str = collection[i].element[j].right;
497             int x = 0;
498             for ( ; x < str.length() ; x++ )
499                 if ( str[x] == CH ) break;
500             if ( x == str.length()-1 )
501                 continue;
502             int y = str[x+1];
503             //cout << "before : " << str << endl;
504             str.erase ( str.begin()+x);
505             str.insert ( str.begin()+x+1 , CH );
506             //cout << "after : " << str << endl;
507             WF cmp = WF ( collection[i].element[j].left , str , -1 , -1 );
508             for ( int k = 0 ; k < m ; k++ )
509             {
510                 bool flag = false;
511                 for ( int t = 0 ; t < collection[k].element.size() ; t++ )
512                 {
513                     if ( cmp == collection[k].element[t] )
514                     {
515                         flag = true;
516                         break;
517                     }
518                 }
519                 if ( flag )
520                 {
521                     go[i][y] = k;
522                 }
523             }
524         }*/
525     for (int t = 0; t < V.size(); t++)

```

```

526     {
527         char ch = V[t];
528         for (int i = 0; i < m; i++)
529         {
530             vector<WF> cmp1;
531             make_cmp(cmp1, i, ch);
532             cout << cmp1.size() << endl;
533             if (cmp1.size() == 0)
534                 continue;
535             for (int j = 0; j < m; j++)
536             {
537                 vector<WF> cmp2;
538                 for (int k = 0; k < collection[j].element.size(); k++)
539                 {
540                     string &str = collection[j].element[k].right;
541                     int x;
542                     for (x = 0; x < str.length(); x++)
543                         if (str[x] == CH)
544                             break;
545                     if (x && str[x - 1] == ch)
546                         cmp2.push_back(WF(collection[j].element[k].left, str,
-1, -1));
547                 }
548                 sort(cmp2.begin(), cmp2.end());
549                 cout << cmp2.size() << endl;
550                 bool flag = true;
551                 if (cmp2.size() != cmp1.size())
552                     continue;
553                 cout << cmp1.size() << endl;
554                 for (int k = 0; k < cmp1.size(); k++)
555                     if (cmp1[k] == cmp2[k])
556                         continue;
557                     else
558                         flag = false;
559                 cout << "out " << endl;
560                 if (flag)
561                     go[i][ch] = j;
562             }
563             // cout << "YES" << endl;
564         }
565     }
566 #ifdef DEBUG
567     puts("-----EDGE-----");
568     stringstream sin;
569     string out;
570     for (int i = 0; i < m; i++)
571         for (int j = 0; j < m; j++)
572             for (int k = 0; k < MAX; k++)
573                 if (go[i][k] == j)
574                 {
575                     sin.clear();
576                     sin << "I" << i << "--" << (char)(k) << "--I" << j;
577                     sin >> out;
578                     printf("%s\n", out.c_str());
579                 }
580 #endif

```

```

581 }
582 bool cmp(char a, char b)
583 {
584     if (a >= 'A' && a <= 'Z' && b >= 'a' && b <= 'z')
585         return 0;
586     if (a >= 'a' && a <= 'z' && b >= 'A' && b <= 'Z')
587         return 1;
588     return a < b;
589 }
590 void make_table()
591 {
592     memset(Goto, -1, sizeof(Goto));
593     int m = collection.size();
594     for (int i = 0; i < m; i++)
595         for (char ch : V) // 优化
596             {
597                 int x = go[i][ch];
598                 if (x == -1)
599                     continue;
600                 if (!isupper(ch))
601                     action[i][ch] = Content(0, x);
602                 else
603                     Goto[i][ch] = x;
604             }
605
606     // 规约
607     for (int i = 0; i < m; i++)
608         for (int j = 0; j < collection[i].element.size(); j++)
609             {
610                 WF &tt = collection[i].element[j];
611                 if (tt.right[tt.right.length() - 1] == CH)
612                     {
613                         if (tt.left[0] == 'S')
614                             action[i]['#'] = Content(2, -1);
615                         else
616                             for (int k = 0; k < V.size(); k++)
617                                 {
618                                     int y = V[k];
619                                     if (!follow[tt.left].count(V[k]))
620                                         continue;
621                                     // 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
622                                     if (action[i][y].type != -1)
623                                         {
624                                             cout << "状态" << i << " " << "发生冲突的符号" <<
625                                             (char)y << endl;
626                                             cout << "不是slr(1)文法" << endl;
627                                         }
628                                     action[i][y] = Content(1, tt.back());
629                                 }
630                     }
631             }
632 #ifdef DEBUG
633     puts("-----SLR(1)分析表-----");

```

```

634     sort(V.begin(), V.end(), cmp);
635     printf("%10s%5c%5s", "|", V[0], "|");
636     for (int i = 1; i < V.size(); i++)
637         printf("%5c%5s", V[i], "|");
638     puts("");
639     for (int i = 0; i < (V.size() + 1) * 10; i++)
640         printf("-");
641     puts("");
642     stringstream sin;
643     for (int i = 0; i < collection.size(); i++)
644     {
645         printf("%5d%5s", i, "|");
646         for (int j = 0; j < V.size(); j++)
647         {
648             char ch = V[j];
649             if (isupper(ch))
650             {
651                 if (Goto[i][ch] == -1)
652                     printf("%10s", "|");
653                 else
654                     printf("%5d%5s", Goto[i][ch], "|");
655             }
656             else
657             {
658                 sin.clear();
659                 if (action[i][ch].type == -1)
660                     printf("%10s", "|");
661                 else
662                 {
663                     Content &temp = action[i][ch];
664                     if (temp.type == 0)
665                         sin << "S";
666                     if (temp.type == 1)
667                         sin << "R";
668                     if (temp.type == 2)
669                         sin << "acc";
670                     if (temp.num != -1)
671                         sin << temp.num;
672                     sin >> temp.out;
673                     printf("%7s%3s", temp.out.c_str(), "|");
674                 }
675             }
676         }
677         puts("");
678     }
679     for (int i = 0; i < (V.size() + 1) * 10; i++)
680         printf("-");
681     puts("");
682 #endif
683 }
684
685 void print(string s1, string s2, string s3, string s4, string s5, string s6,
686           string s7)
687 {
688     printf("%-15s|%-15s%-15s%-20s|%-15s%-15s%-15s\n", s1.c_str(), s2.c_str(),
689           s3.c_str(), s4.c_str(), s5.c_str(),

```

```

688         s6.c_str(), s7.c_str());
689     }
690
691     string get_steps(int x)
692     {
693         stringstream sin;
694         sin << x;
695         string ret;
696         sin >> ret;
697         return ret;
698     }
699
700     template <class T>
701     string get_stk(vector<T> stk)
702     {
703         stringstream sin;
704         for (int i = 0; i < stk.size(); i++)
705             sin << stk[i];
706         string ret;
707         sin >> ret;
708         return ret;
709     }
710
711     string get_shift(WF &temp)
712     {
713         stringstream sin;
714         sin << "reduce(" << temp.left << "->" << temp.right << ")";
715         string out;
716         sin >> out;
717         return out;
718     }
719
720     void analyse(string src)
721     {
722         std::ofstream outfile("4yuan.txt");
723         print("steps", "op-stack", "input", "operation", "state-stack", "ACTION",
724             "GOTO");
725         vector<char> op_stack;
726         vector<int> st_stack;
727         vector<string> symbol2;
728         string label;
729         auto it = symbol.begin();
730         int n = 1;
731         src += "#";
732         op_stack.push_back('#');
733         st_stack.push_back(0);
734         int steps = 1;
735         for (int i = 0; i < src.length(); i++)
736         {
737             char u = src[i];
738             int top = st_stack[st_stack.size() - 1];
739             // action两个参数: 状态集合编号, 输入符号
740             Content &act = action[top][u];
741             // cout << "YES : " << i << " " << u << " " << top << " " << act.type
742             << endl;
743             if (act.type == 0)

```



```

742     {
743         print(get_steps(steps++), get_stk(op_stack), src.substr(i),
"shift", get_stk(st_stack), act.out, "");
744         op_stack.push_back(u);
745         st_stack.push_back(act.num);
746         if (u == 'i')
747         {
748             symbol2.push_back(*it);
749             ++it;
750         }
751     }
752     else if (act.type == 1)
753     {
754         // act.num是当前规约的产生式编号
755         string s1;
756         string s2;
757         switch (act.num)
758         {
759             case 9:
760                 // cout << "E->E+T" << endl;
761                 // cout << "symbol2.size:" << symbol2.size() << endl;
762                 s1 = symbol2.back();
763                 symbol2.pop_back();
764                 s2 = symbol2.back();
765                 symbol2.pop_back();
766                 label = "L" + std::to_string(n);
767                 n++;
768                 symbol2.push_back(label);
769                 outfile << "+" << s2 << ' ' << s1 << ' ' << label << endl;
770                 break;
771             case 10:
772                 // cout << "E->E-T" << endl;
773                 // cout << "symbol2.size:" << symbol2.size() << endl;
774                 s1 = symbol2.back();
775                 symbol2.pop_back();
776                 s2 = symbol2.back();
777                 symbol2.pop_back();
778                 label = "L" + std::to_string(n);
779                 n++;
780                 symbol2.push_back(label);
781                 outfile << "-" << s2 << ' ' << s1 << ' ' << label << endl;
782                 break;
783                 break;
784             case 12:
785                 // cout << "T->T*F" << endl;
786                 // cout << "symbol2.size:" << symbol2.size() << endl;
787                 s1 = symbol2.back();
788                 symbol2.pop_back();
789                 s2 = symbol2.back();
790                 symbol2.pop_back();
791                 label = "L" + std::to_string(n);
792                 n++;
793                 symbol2.push_back(label);
794                 outfile << "*" << s2 << ' ' << s1 << ' ' << label << endl;
795                 break;
796             case 13:

```

```

797         // cout << "T->E/F" << endl;
798         // cout << "symbol2.size:" << symbol2.size() << endl;
799         s1 = symbol2.back();
800         symbol2.pop_back();
801         s2 = symbol2.back();
802         symbol2.pop_back();
803         label = "L" + std::to_string(n);
804         n++;
805         symbol2.push_back(label);
806         outfile << "/" << s2 << " " << s1 << " " << label << endl;
807         break;
808     case 7: // 赋值
809         // cout << "E=ID" << endl;
810         // cout << "symbol2.size:" << symbol2.size() << endl;
811         s1 = symbol2.back();
812         symbol2.pop_back();
813         s2 = symbol2.back();
814         symbol2.pop_back();
815         outfile << " := " << s1 << " " << "_ " << s2 << endl;
816         break;
817     case 5: // if
818         // cout << "if" << endl;
819         // cout << "symbol2.size:" << symbol2.size() << endl;
820         s1 = symbol2.back();
821         outfile << "jnz " << s1 << " _ 0" << endl;
822         break;
823     case 6: // else
824         // cout << "else" << endl;
825         // cout << "symbol2.size:" << symbol2.size() << endl;
826         outfile << "j _ _ 0" << endl;
827         break;
828     case 4: // while
829         // cout << "while" << endl;
830         // cout << "symbol2.size:" << symbol2.size() << endl;
831         s1 = symbol2.back();
832         outfile << "jnz " << s1 << " _ 0" << endl;
833         break;
834     case 3: // do
835         outfile << "j _ _ 0" << endl;
836         break;
837     default:
838         break;
839 }
840 WF &tt = wf[act.num];
841 int y = st_stack[st_stack.size() - tt.right.length() - 1];
842 int x = Goto[y][tt.left[0]];
843 // cout << y << " " << tt.left[0] << " " << x << endl;
844 print(get_steps(steps++), get_stk(op_stack), src.substr(i),
get_shift(tt), get_stk(st_stack), act.out, get_steps(x));
845 for (int j = 0; j < tt.right.length(); j++)
846 {
847     st_stack.pop_back();
848     op_stack.pop_back();
849 }
850 op_stack.push_back(tt.left[0]);
851 st_stack.push_back(x);

```

```

852         i--;
853     }
854     else if (act.type == 2)
855     {
856         print(get_steps(steps++), get_stk(op_stack), src.substr(i),
857 "Accept", get_stk(st_stack), act.out, "");
858         // i--;
859     }
860     else
861         continue;
862 }
863 string toSymbol()
864 {
865     std::ifstream infile("input.txt");
866     std::string line = "";
867     std::string word = "";
868
869     if (infile.is_open())
870     {
871         while (infile >> word)
872         {
873             cout << word << endl;
874             if (word == "IFTK")
875             {
876                 line += "f";
877                 infile >> word;
878                 continue;
879             }
880             else if (word == "THENTK")
881             {
882                 line += "t";
883                 infile >> word;
884                 continue;
885             }
886             else if (word == "ELSETK")
887             {
888                 line += "e";
889                 infile >> word;
890                 continue;
891             }
892             else if (word == "IDENFR" || word == "INTCON")
893             {
894                 line += 'i';
895                 infile >> word;
896                 symbol.push_back(word);
897                 continue;
898             }
899             else if (word == "PLUSTK")
900             {
901                 line += "+";
902                 infile >> word;
903                 continue;
904             }
905             else if (word == "MULTITK")
906             {

```

```
907         line += "*";
908         infile >> word;
909         continue;
910     }
911     else if (word == "DIVTK")
912     {
913         line += "/";
914         infile >> word;
915         continue;
916     }
917     else if (word == "MINUTETK")
918     {
919         line += "-";
920         infile >> word;
921         continue;
922     }
923     else if (word == "LPARTK")
924     {
925         line += "(";
926         infile >> word;
927         continue;
928     }
929     else if (word == "RPARTK")
930     {
931         line += ")";
932         infile >> word;
933         continue;
934     }
935     else if (word == "ASSIGNTK")
936     {
937         line += "d";
938         infile >> word;
939         continue;
940     }
941     else if (word == "WHILETK")
942     {
943         line += "w";
944         infile >> word;
945         continue;
946     }
947     else if (word == "DOTK")
948     {
949         line += "o";
950         infile >> word;
951         continue;
952     }
953     else if (word == "LBRATK")
954     {
955         line += "{";
956         infile >> word;
957         continue;
958     }
959     else if (word == "RBRATK")
960     {
961         line += "}";
962         infile >> word;
```

```

963             continue;
964         }
965         else
966         {
967             infile >> word;
968             continue;
969         }
970     }
971     infile.close();
972 }
973 else
974 {
975     std::cout << "Unable to open file";
976 }
977 cout << "symbol.size:" << symbol.size() << endl;
978 return line;
979 }
980 int main()
981 {
982     int n;
983     char s[MAX];
984     while (~scanf("%d", &n))
985     {
986         for (int i = 0; i < n; i++)
987         {
988             scanf("%s", s);
989             int len = strlen(s), j;
990             for (j = 0; j < len; j++)
991                 if (s[j] == '-')
992                     break;
993             s[j] = 0;
994             wf.push_back(WF(s, s + j + 2, -1, -1));
995 #ifdef DEBUG
996             wf[wf.size() - 1].print();
997 #endif
998         }
999         make_item();
1000         make_first();
1001         make_follow();
1002         make_set();
1003         make_V();
1004         make_go();
1005         make_table();
1006         string str = toSymbol();
1007         cout << str << endl;
1008         analyse(str);
1009     }
1010 }

```