中山大学数据科学与计算机学院本科生实验报告

课程名称：编译器构造实验 任课教师：陈炬桦 教学助理（TA）：

|  |  |  |  |
| --- | --- | --- | --- |
| 学年度 | 2018-2019 | 学期 | 第二学期 |
| 年级 | 2016级 | 专业（方向） | 计算机科学与技术 |
| 学号 | 16337233 | 姓名 | 王凯祺 |
| 电话 | 13434386182 | Email | [Wangkq3@mail2.sysu.edu.cn](mailto:Wangkq3@mail2.sysu.edu.cn) |
| 开始日期 | 2019/3/15 | 完成日期 | 2019/3/15 |

1. 实验题目

**1.1 Description**

输入开始符号，非终结符，终结符，产生式  
压缩自产生式文法和不可达文法后，按非终结符顺序输出产生式；

**1.2 Input**

 输入开始符号；  
非终结符个数，非终结符，空格符分隔；  
终结符个数，终结符，空格符分隔；  
产生式的个数，各产生式的左边和右边符号，空格符分隔；

**1.3 Output**

delete self production:自产生式文法  
unreached Vn:不可达非终结符  
delete production:不可达产生式  
delete VN:不可达非终结符  
G[开始符号]：  
压缩自产生式文法和不可达文法后，按非终结符顺序输出各产生式；

2. 算法描述(介绍程序模块功能；流程图)

(1) 在输入产生式时，先判断产生式左边和右边是否相等，若相等则删除。

(2) 将产生式的左边的非终结符到右边的非终结符/终结符连一条有向边。

(3) 从开始符号G开始，对整张图做一次遍历。

(4) 遍历不到的节点记为不可达节点，首先将以它为左边的产生式删除，再将该节点删除。

(5) 按非终结符顺序输出产生式。

3. 测试数据

样例输入

Z

8 Z E F P G T Q S

3 + \* i

18

Z E+T

E E

P G

F F

P G

G G

T T\*i

Q E

S i

E S+F

F FP

G GG

Q E+F

E T

F P

G F

Q T

Q S

样例输出

delete self production:E::=E

delete self production:F::=F

delete self production:G::=G

unreached Vn:Q

delete production:Q::=E

delete production:Q::=E+F

delete production:Q::=T

delete production:Q::=S

delete VN:Q

G[Z]:

Z::=E+T

E::=S+F | T

F::=FP | P

P::=G | G

G::=GG | F

T::=T\*i

S::=i

4. 程序清单

#include <bits/stdc++.h>

using namespace std;

string S;

vector<string> Vn;

vector< vector<string> > ans;

vector<string> Vt;

vector< pair<string, string> > t;

bool mp[256];

int main() {

cin >> S;

mp[S[0]] = 1;

int n; cin >> n;

ans.resize(n);

for (int i = 1; i <= n; ++i) {

string x;

cin >> x;

Vn.push\_back(x);

}

cin >> n;

for (int i = 1; i <= n; ++i) {

string x;

cin >> x;

Vt.push\_back(x);

}

cin >> n;

while (n--) {

string x, y;

cin >> x >> y;

if (x == y) {

printf("delete self production:%s::=%s\n", x.c\_str(), y.c\_str());

continue;

}

t.push\_back(make\_pair(x, y));

}

int cnt = 1;

while (cnt) {

cnt = 0;

for (int i = 0; i < (int)t.size(); ++i) {

if (mp[t[i].first[0]]) {

for (int j = 0; j < (int)t[i].second.length(); ++j)

if (!mp[t[i].second[j]]) {

mp[t[i].second[j]] = 1;

++ cnt;

}

}

}

}

for (int i = 0; i < Vn.size(); ++i)

if (!mp[Vn[i][0]]) {

printf("unreached Vn:%s\n", Vn[i].c\_str());

for (int j = 0; j < (int)t.size(); ++j)

if (Vn[i] == t[j].first)

printf("delete production:%s::=%s\n", t[j].first.c\_str(), t[j].second.c\_str());

printf("delete VN:%s\n", Vn[i].c\_str());

}

for (int i = 0; i < (int)t.size(); ++i) {

int x = -1;

for (int j = 0; j < (int)Vn.size(); ++j)

if (t[i].first == Vn[j]) {

x = j;

break;

}

ans[x].push\_back(t[i].second);

}

printf("G[%s]:\n", S.c\_str());

for (int i = 0; i < (int)Vn.size(); ++i)

if (mp[Vn[i][0]]) {

printf("%s::=%s", Vn[i].c\_str(), ans[i][0].c\_str());

for (int j = 1; j < (int)ans[i].size(); ++j)

printf(" | %s", ans[i][j].c\_str());

puts("");

}

}