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 | 专业（方向） | 计算机科学与技术 |
| 学号 | 16337052 | 姓名 | 杜尔鑫 |
| 电话 | 15626281204 | Email | duerx@mail2.sysu.edu.cn |
| 开始日期 | 2019-03-15 | 完成日期 | 2019-03-15 |

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；

此题需要提交实验报告；“实验报告用“学号+姓名+31”

**1.2 Input**

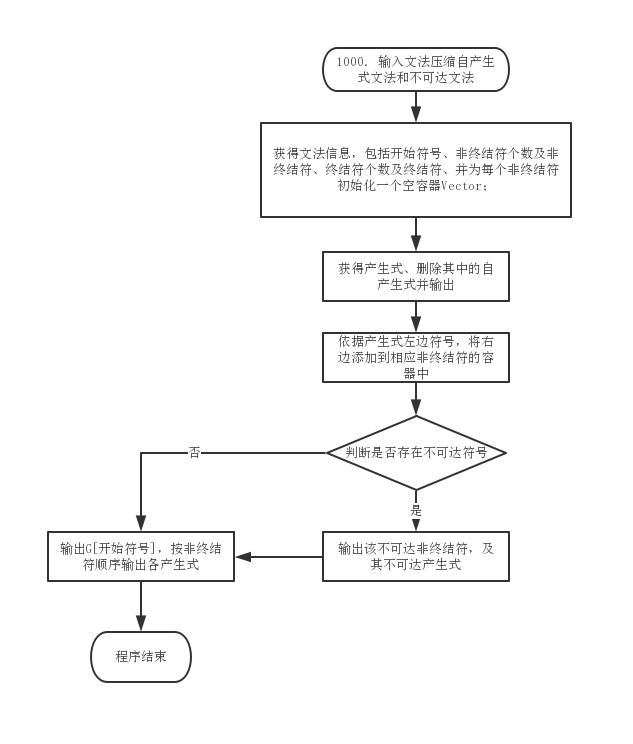
输入开始符号；  
非终结符个数，非终结符，空格符分隔；  
终结符个数，终结符，空格符分隔；  
产生式的个数，各产生式的左边和右边符号，空格符分隔；

**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. 输入文法信息并储存，包括开始符号、非终结符个数及非终结符、终结符个数及终结符；并为每个非终结符初始化一个空容器；
2. 输入产生式的个数，各产生式的左边和右边符号；压缩自产生式文法，然后根据各产生式左边将右边添加到相应非终结符的表达式容器中；
3. 判断是否存在不可达非终结符。如果有进入4)，无进入5)
4. 输出不可达非终结符，及其不可达产生式，压缩不可达文法；
5. 输出G[开始符号]，按非终结符顺序输出各产生式；



3. 测试数据(2组)

**3.1 Standard Example**

**3.1.1 Input**

|  |
| --- |
| 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 |

**3.1.2 Output**

|  |
| --- |
| 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 |

**3.2 Addition Test**

**3.2.1 Input**

|  |
| --- |
| E  4 E T F G  7 + - \* / ( ) i  11  E E+T  T T\*F  G F/T  E T  T F  F (E)  F i  E E-T  T T/F  E E  G E\*F |

**3.2.2 Output**

|  |
| --- |
| delete self production:E::=E  unreached Vn:G  delete production:G::=F/T  delete production:G::=E\*F  delete VN:G  G[E]:  E::=E+T | T | E-T  T::=T\*F | F | T/F  F::=(E) | i |

4. 程序清单

|  |
| --- |
| #include <iostream>  #include <vector>  #include <algorithm>  using namespace std;  struct Vns\_Struct { int Nv; string VN[10];} ;  struct Vts\_Struct { int Nt; string VT[10];} ;  struct ps\_Struct { int Np; string PL[20], PR[20];} ;  string S;  int main(int argc, char const \*argv[])  {      Vns\_Struct Vns;      Vts\_Struct Vts;      ps\_Struct ps;      std::vector<string> reached;      // Begin Symbol      cin >> S;      reached.push\_back(S);      // Begin Nonterminal      cin >> Vns.Nv;      std::vector<string> out[Vns.Nv];      for (int i = 0; i < Vns.Nv; ++i)          cin >> Vns.VN[i];      // Begin Terminal      cin >> Vts.Nt;      for (int i = 0; i < Vts.Nt; ++i)          cin >> Vts.VT[i];      // Input Productions      // And delete self production      cin >> ps.Np;      for (int i = 0; i < ps.Np; ++i)      {          cin >> ps.PL[i] >> ps.PR[i];          reached.push\_back(ps.PR[i]);          if (ps.PL[i] == ps.PR[i])              cout << "delete self production:" << ps.PL[i] << "::=" << ps.PR[i] << endl;          else          {              int j = 0;              for (; j < Vns.Nv; ++j)                  if (Vns.VN[j] == ps.PL[i])                      break;              out[j].push\_back(ps.PR[i]);          }      }      // Find the unreached Vn      // And delete production and VN      for (int i = 0; i < Vns.Nv; ++i)      {          vector<string>::iterator find\_unreached;          find\_unreached = std::find(reached.begin(), reached.end(), Vns.VN[i]);          if (find\_unreached == reached.end())          {              cout << "unreached Vn:" << Vns.VN[i] << endl;              int j = 0;              for (; j < Vns.Nv; ++j)                  if (Vns.VN[j] == Vns.VN[i])                      break;              for (vector<string>::iterator iter\_i = out[j].begin(); iter\_i != out[j].end(); ++iter\_i)                  cout << "delete production:" << Vns.VN[i] << "::=" << \*iter\_i << endl;              cout << "delete VN:" << Vns.VN[i] << endl;              out[j].clear();          }      }      // Begin Print Productions      cout << "G[" << S << "]:" << endl;      for (int i = 0; i < Vns.Nv; ++i)      {          if (out[i].empty())              continue;          cout << Vns.VN[i] << "::=";          for (vector<string>::iterator iter\_i = out[i].begin(); iter\_i != out[i].end(); ++iter\_i)          {              cout << \*iter\_i;              if (iter\_i+1 == out[i].end())                  cout << endl;              else                  cout << " | ";          }      }      return 0;  } |