**C++中的STL模板库中的关联容器map和set介绍：**

map是键值（key）和属性值（value）的一个映射。其中key和value可以选择任意数据类型。

1. 创建一个利用string作为key的map对象，string与int一一对应：

map <string,int> mp;

1. 向map中添加元素：

mp[“Anna”]=1;//理解为叫Anna这个学生对应学号1

mp[“Beta”]=2;//理解为叫Beta这个学生对应学号2

1. 查找key出现的次数：

cout<<mp.count(“Anna”); // 输出1

找到“Beta”的位置：

auto iter = mp.find(“Beta”);cout<<\*iter;

1. 利用迭代器进行遍历：

for(auto it = mp.begin();it != mp.end();it++)

printf("%s->%d\n",it->first,it->second);

在本次实验中，采用了map<char, set<string> > P来存储所有产生式。非终结符char与其相应的string集合对应。

set也是关联容器，它区别于map，仅存储所有的key。

1）创建一个set。

set <string> str\_set;

2）向set里面插入元素

str\_set.insert(“E+T”);

str\_set.insert(“T”);

P[‘E’]=str\_set;//E对应于它的产生式

1. **看懂parser\_my.cpp的main中的程序:**

string filename\_gramer = "D:\\parse\_test1.txt";

Grammar \*grammar=new Grammar(filename\_gramer);，

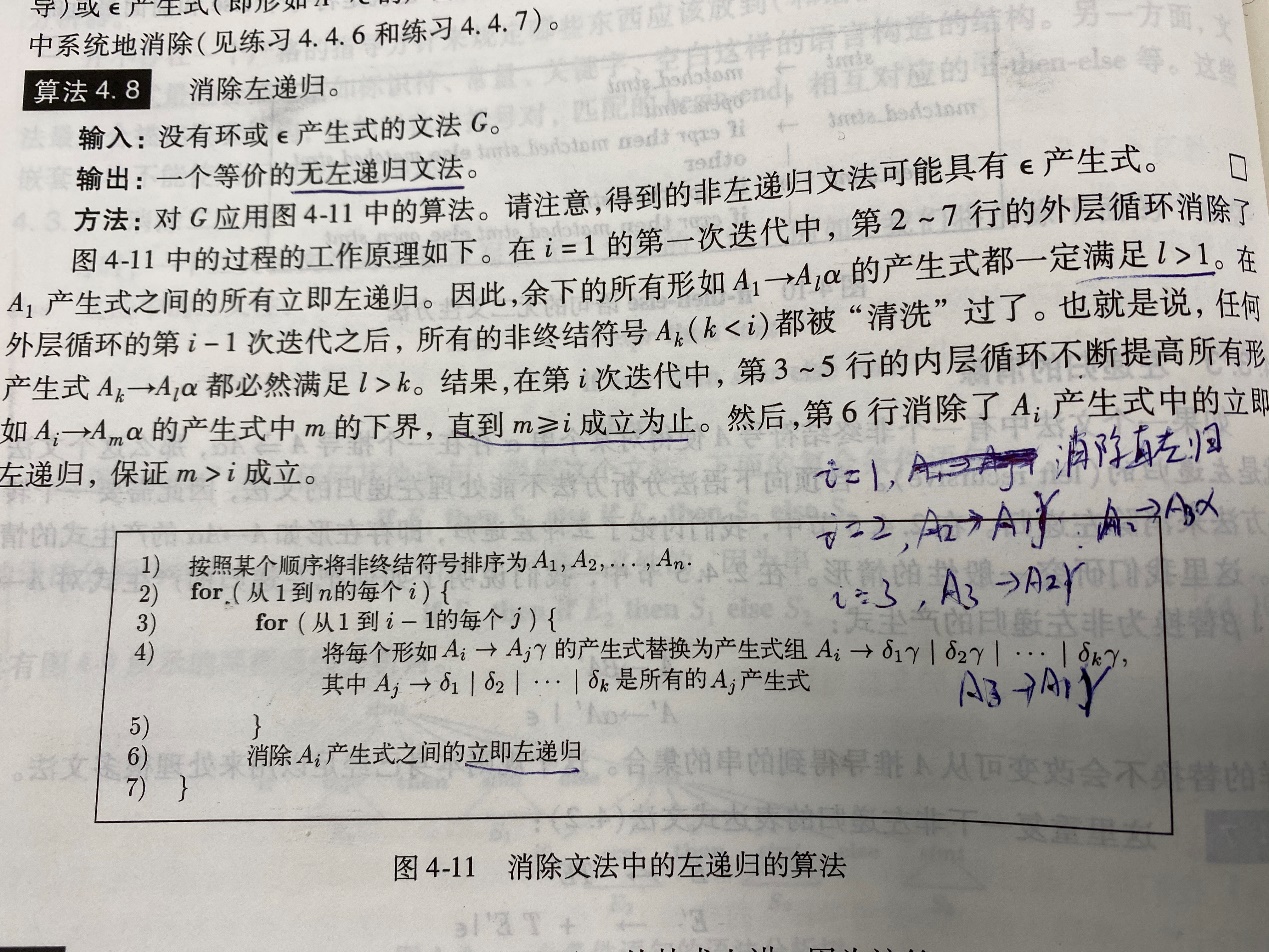
利用P读入所有产生式。当前文法为：

E->T|E+T;

T->F|T\*F;

F->i|(E);

**2、学习消除左递归算法如下：**



（方框内）首先把所有非终结符进行排序，当前文法有3个非终结符E、T和F，即按此排序。然后2层for循环如下：

for i=1:n

{

for j=1:i-1

{

代入替换；

}

消除Ai的直接左递归。

}

i=1时，j无取值。内层循环不执行，消除A1的直接左递归，即消除E的直接左递归，对应于代码的remove\_left\_gene(char c) 函数，c对应于E。看懂如何消除直接左递归。

例如：在当前文法中，消除E的直接左递归：

把E->E+T|T改为

E->TA;

E->+TA|@; //@表示空串，其中A是新添加的非终结符。

**3、继续理解消除左递归算法：**

i=2时，j=1。查找所有A2->A1r的产生式 (r为任意符号串)，把所有A1作为左部的产生式进行代入，使得A2->Akr，其中k>=2。

消除A2的直接左递归，那么所有A2->Akr，其中k>2。

i=3时，j=2。查找所有A3->A1r或A3->A2r的产生式，替换后使得A3->Akr，其中k>=3。

消除A3的直接左递归，那么所有A3->Akr，其中k>3。

这样一直下去，可以保证所有Am->Akr，其中k>m，产生式右部的第一个符号一定排在左部第一个符号的后面，这样就消除了所有间接的左递归。对应于代码的remove\_left\_recursion()函数，需要自己实现。

例如：在当前文法中，i=3时，T->F的F需要代入替换。

**4、实现First和Follow集的计算：**

