

学生学号	0121710880415	成绩	
------	---------------	----	--

武汉理工大学

学生实验报告书

实验课程名称	编译原理
开 课 学 院	计算机科学与技术学院
指导教师姓名	王云华
学 生 姓 名	颜道江
学生专业班级	软件 1704

2019 -- 2020 学 年 第 二 学 期

实验课程名称： 编译原理

实验项目名称	语法分析			实验成绩	
实验者	颜道江	专业班级	软件 1704	组别	
同组者				实验日期	2019/12/22

第一部分：实验内容描述

1. 设计题目

集合 FIRSTVT(P) 构造算法的程序实现。

2. 设计目的及设计要求

构造一程序，实现 LASTVT(P) 集合的构造算法。对任一给定的算符文法 G，程序输出所有非终结符 P 的 LASTVT(P)。

3. 设计内容

3.1 算法设计

对于 FIRSTVT 集的构造可以给出一个算法，算法基于下面的两条规则：

1. 若有产生式 $A \rightarrow a \cdots$ 或 $A \rightarrow Ba \cdots$ ，则 $a \in \text{FIRSTVT}(A)$ ，其中 A、B 为非终结符，a 为终结符；
2. 若 $a \in \text{FIRSTVT}(B)$ 且有产生式 $A \rightarrow B \cdots$ ，则有 $a \in \text{FIRSTVT}(A)$ 。

3.2 详细设计与实现

1. 根据文件中输入的文法来提取所有的终结符和非终结符，具体的提取方法是：根据“ \rightarrow ”来对产生式的左右两端来进行分割，分割得到的左边必然是非终结符，得到的右边必然是终结符，分割完成得到终结符集 VT 和非终结符集和 VN。
2. 构造一个两层的产生式字典，其中字典的外层键为产生式的编号，值为一个产生式字典，内层的产生式字典“键”为“left”和“right”，值为对应分割得到的产生式的左边和右边；
3. 具体的求解过程如下步骤所示：
 - 3.1 将所有非终结符对应的 FIRSTVT 初始化为空，即为一个空字典；

- 3.2 对所有的产生式进行一遍完整的扫描，扫描的过程中对算法 3.1 中描述的 $A \rightarrow a \cdots$ 或 $A \rightarrow Ba \cdots$ 进行判断，如果是符合这种形式的向初始化的字典中进行添加，其中产生式的左侧就是那个对应非终结符；
- 3.3 为了对算法 3.1 中描述的情况 2 进行求解，首先初始化一个空栈，并且按照，将 3.2 中已经求得的非终结符入栈；
- 3.4 将栈顶的 (B, a) 弹出
- 3.5 检查所有的产生式，如果存在 $A \rightarrow B \cdots$ ，并且 $\text{firstvt}(B)$ 又求出来的情况；那么 $a \in \text{FIRSTVT}(A)$ ，并将 (A, a) 推进栈；
- 3.6 重复 3.4 和 3.5 知道栈为空为止。

4. 输入输出设计

本次的实验中输入文件尾 Grammar.txt, 其中对按行进行了详细的设计，具体的输入文件结构如下图所示：

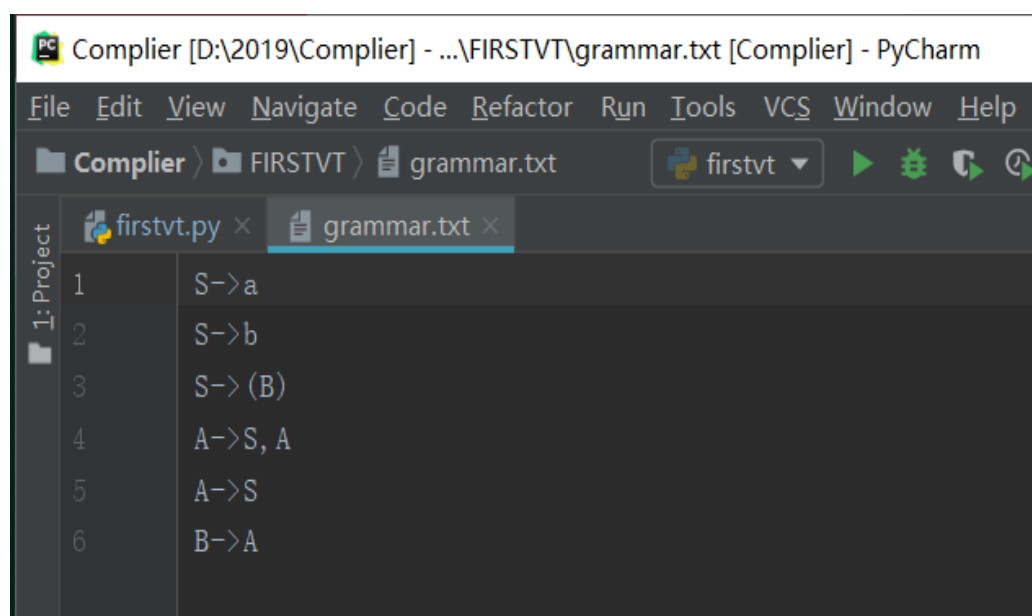


图 输入文件详细结构

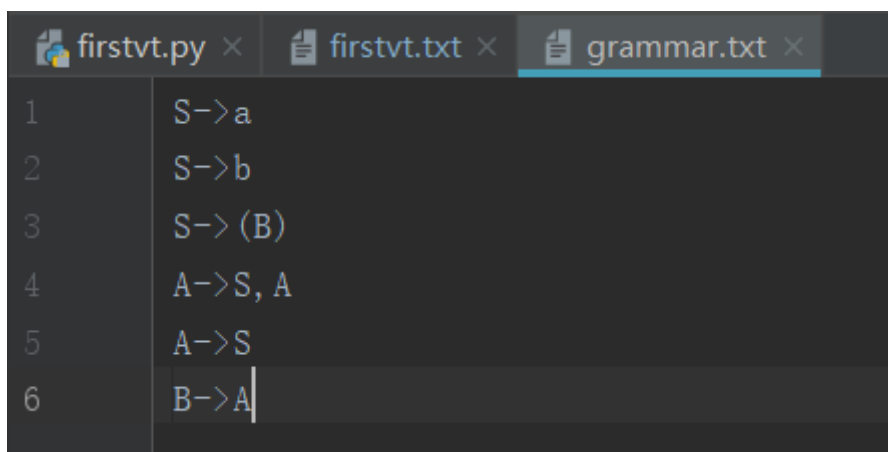
在进行文件读取的过程中需要进行一些预处理工作，主要宝行下面的几个方面：

1. 按行进行产生式的读取；
2. 读入产生式的过程中需要去掉换行符 “\n” 等其他一些没用的符号；
3. 读取的产生式进行预处理的时候，通过 “->” 进行分割；

程序运行得到的结果包括控制台显示输出和写进 txt 文件。

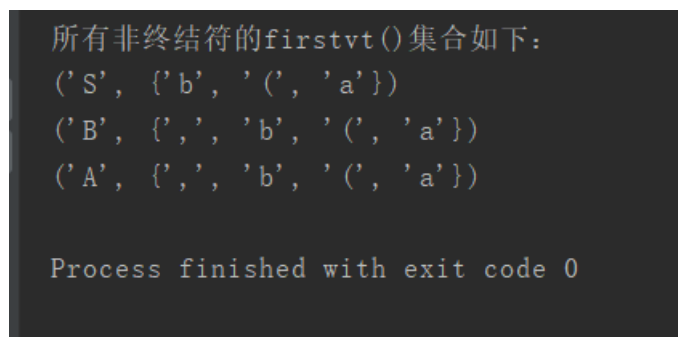
5. 运行结果

程序运行选择输入的的产生式文件如下所示：



```
firstvt.py × firstvt.txt × grammar.txt ×
1 S->a
2 S->b
3 S->(B)
4 A->S, A
5 A->S
6 B->A
```

求得的各个非终结符的 firstvt 集合如下：



```
所有非终结符的firstvt()集合如下：
('S', {'b', '(', 'a'})
('B', {'', 'b', '(', 'a'})
('A', {'', 'b', '(', 'a'})

Process finished with exit code 0
```

6. 总结

本次的实验是求 FIRSTVT(P), 相对于上次的实验本次的实验难度更简单一些, 主要是因为本次的实验中需要实现的算法更少, 算法的步骤也更加的简单。通过本次的实验也将 FIRSTVT 集进行了一次完整的复习, 之前在学习的过程中, 对于集合的求解没有通过程序来进行实现, 主要进行通过观察来进行求解的, 通过本次的实验掌握了求解的步骤, 以后在做题的过程中也可以运用来进行检查。

7. 附录——核心代码

```
1. # -*- coding:utf-8 -*-
2. # file: firstvt.py
3. class Grammar(object):
4.     """
5.     根据算符优先文法求 FIRSTVT(), 非终结符为大写字母 A-Z, 其他全为小写字母
6.     """
7.
8.     def __init__(self, filename):
9.         self productions = {}
10.        self productions_num = 0
11.        # 终结符集合
12.        self.VT = set()
13.        # 非终结符集合
14.        self.VN = set()
15.
16.        self.initGrammar(filename)
17.
18.    def initGrammar(self, filename):
19.        fin = open(filename, 'rt')
20.        productions = []
21.        while True:
22.            line = fin.readline()
23.            if not line:
24.                break
25.            productions.append(line)
26.            # 去掉换行符
27.            productions = [p.rstrip("\n") for p in productions]
28.            for p in productions:
29.                production = {}
30.                # 根据“->”分割一条产生式的左右两端
31.                a = p.split("->")
32.                production["left"] = a[0]
33.                production["right"] = a[1]
34.                self productions_num += 1
35.                self productions[self productions_num] = production
36.                # 将产生式左边符号添加到非终结符集合中
37.                if a[0] not in self.VN:
38.                    self.VN.add(a[0])
39.                # 将产生式右边的非终结符添加到非终结符集合中
40.                for ch in a[1]:
41.                    if (not 'A' <= ch <= 'Z') and (ch not in self.VT):
```

```

42.         self.VT.add(ch)
43.
44.     def show_info(self):
45.         print("所有产生式: ", self productions)
46.         print("所有终结符: ", self.VT)
47.         print("所有非终结符: ", self.VN)
48.
49.     def generate_firstvt(self):
50.         firstvt = {}
51.         # 先全初始化为空
52.         for ch in self.VN:
53.             firstvt[ch] = set({})
54.
55.         # 第一遍先扫描形如 A->a...或 A->Ba...的产生式
56.         for p_i in range(1, self productions_num + 1):
57.             # A->a...形式
58.             if self productions[p_i]["right"][0] in self.VT:
59.                 firstvt[self productions[p_i]["left"]].add(self productions[p_i
60. ]["right"][0])
61.             # A->Ba...形式
62.             elif len(self productions[p_i]["right"]) >= 2 \
63.                 and self productions[p_i]["right"][0] in self.VN \
64.                 and self productions[p_i]["right"][1] in self.VT:
65.                 firstvt[self productions[p_i]["left"]].add(self productions[p_i
66. ]["right"][1])
67.         # 第二遍再扫描形如 A->B.. , B->b...或 B->Cb...的产生式
68.         # 这样只要将 firstvt(B)加入到 firstvt(A)中
69.         # 如出现 E->T , T->F , F->P 这样的情况, 那么执行的顺序不同可能会产生不同的结
70.         # 因此采用一个栈来完成
71.         stack = []
72.
73.         # 将当前所有非终结符的 firstvt 入栈
74.         for key in firstvt.keys():
75.             for item in firstvt[key]:
76.                 stack.append((key, item))
77.
78.         while len(stack) > 0:
79.             # 弹出当前栈顶, 二元组为 ==> (终结符, 终结符 firstvt 中的一个)
80.             key, item = stack.pop()
81.             for p_i in range(1, self productions_num + 1):
82.                 # 如果存在 A->B.., 并且 firstvt(B)又求出来了的情况

```

```
83.         if self productions[p_i]["right"][0] == key \
84.             and self productions[p_i]["left"] != key \
85.             and item not in firstvt[self productions[p_i]["left"]]:

86.             firstvt[self productions[p_i]["left"]].add(item)
87.             stack.append((self productions[p_i]["left"], item))
88.
89.     return firstvt
90.
91. def outputToFile(self, filename, firstvt):
92.     """将计算得到的结果保存到文件中"""
93.     with open(filename, "w") as f:
94.         f.write("firstvt 集合如下:")
95.         for key, value in firstvt.items():
96.             f.write("\n" + key + ":")
97.             # 去掉单引号
98.             f.write "[" + ",".join(value) + "]"
99.
100.
101. if __name__ == '__main__':
102.     grammar = Grammar("grammar.txt")
103.
104.     # print("该算符优先文法的信息如下: ")
105.     # grammar.show_info()
106.
107.     print("所有非终结符的 firstvt()集合如下: ")
108.     firstvt = grammar.generate_firstvt()
109.     [print(item) for item in firstvt.items()]
110.
111.     grammar.outputToFile("firstvt.txt", firstvt)
```