**编译原理实验报告**

**实验名称：词法分析实验**

实验目的和要求

通过设计一个FIRST集合计算器，帮助理解FIRST集合的计算方法。FIRST集合是语法分析中的重要概念，掌握其计算对后续的语法分析有重要意义。

**一、实验内容**

实验内容如下：

1. 设计一个基于Tkinter的GUI程序，输入文法产生式，计算并显示FIRST集合。
2. 根据输入的文法产生式，对每个非终结符计算FIRST集合。
3. 实现避免循环依赖的算法。
4. 结果展示在GUI界面上。

**二、实验步骤**

实验步骤如下：

1. 创建图形用户界面
   1. 创建主窗口：

使用Tkinter库创建一个主窗口，窗口标题为“FIRST集合计算器”。

* 1. 创建输入标签和文本框：

添加一个标签，提示用户输入文法的产生式。

创建一个多行文本框，用于用户输入文法的产生式。

* 1. 创建计算按钮：

添加一个按钮，显示文本为“计算FIRST集合”，点击按钮时触发计算FIRST集合的操作。

* 1. 创建结果显示框：

添加一个文本框，用于显示计算结果。设置为只读状态。

1. 计算FIRST集合
   1. 解析用户输入：

获取用户在文本框中输入的文法产生式，解析为字典形式。

* 1. 计算FIRST集合：

对每个非终结符进行如下操作：

初始化一个空集合作为其FIRST集合。

遍历该非终结符的所有产生式：

若产生式以终结符或ε开始，则将其加入到FIRST集合中。

若产生式以非终结符开始，则递归计算该非终结符的FIRST集合。

若产生式的首符号无法推导出ε，则将其加入到FIRST集合中。

若该非终结符的所有产生式均可推导出ε，则将ε加入到其FIRST集合中。

* 1. 显示计算结果：

将计算得到的FIRST集合显示在结果显示框中。

1. 主程序入口
   1. 定义终结符集合：

定义文法中可能出现的终结符集合，包括运算符、分界符等。

* 1. 调用函数创建并显示窗口：

调用 create\_window() 函数，创建并显示图形用户界面。

1. 运行实验
   1. 输入文法产生式：

在输入文本框中输入文法的产生式，每行一个产生式，格式为“非终结符->产生式|产生式”。

* 1. 计算FIRST集合：

点击“计算FIRST集合”按钮，程序将会计算输入文法的FIRST集合，并将结果显示在结果显示框中。

1. 分析代码实现
   1. 创建窗口和界面元素：

使用Tkinter库创建主窗口，标签、文本框、按钮和结果显示框等界面元素，设置它们的属性和布局。

* 1. 计算FIRST集合的算法：

实现 calculate\_first\_in(productions) 函数，该函数接收一个字典 productions，其中键为非终结符，值为对应的产生式列表。

在该函数中，首先初始化一个字典 firsts 用于保存各非终结符的FIRST集合，以及一个字典 completed 用于记录已经计算过的非终结符。

然后，定义内部函数 calculate\_first(non\_terminal)，用于递归计算给定非终结符的FIRST集合。该函数使用栈来处理产生式中的非终结符，并通过循环遍历产生式，根据不同情况进行处理，直至所有相关非终结符的FIRST集合都被计算出来。

最后，对给定文法的所有非终结符，调用 calculate\_first() 函数计算其FIRST集合，并返回完整的FIRST集合字典。

* 1. 处理用户输入：

在 process\_input() 函数中，解析用户在文本框中输入的文法产生式，并调用 calculate\_first\_in() 函数计算FIRST集合。

将计算结果格式化成字符串，并显示在结果显示框中。

1. 实验结果与讨论
   1. 输入测试样例：

输入不同的文法产生式，测试程序对不同文法的FIRST集合计算是否正确。

* 1. 输出结果分析：

分析程序输出的FIRST集合是否与预期结果一致，检查是否存在计算错误或逻辑漏洞。

* 1. 优化改进：

如果发现程序存在问题或性能瓶颈，可以对代码进行优化改进，提高计算效率或修复错误。

* 1. 扩展功能：

根据实际需求，可以扩展程序功能，如计算FOLLOW集合、构造LL(1)分析表等。

* 1. 讨论结果：

讨论实验结果的合理性和可行性，总结实验过程中遇到的问题和解决方法，以及对算法和程序的改进建议。

通过以上步骤，可以全面了解实验的设计思路、算法实现和结果分析，进而深入掌握FIRST集合的计算方法及相关编程技巧。

实验代码如下：

import tkinter as tk

from tkinter import scrolledtext

# 创建并显示窗口

def create\_window():

# 创建主窗口

root = tk.Tk()

root.title("FIRST集合计算器")

# 创建并设置输入标签

input\_label = tk.Label(root,

text="请输入文法的产生式(每行一个产生式，格式:非终结符->产生式|产生式，点击按钮进行计算)：",

fg="blue", font=("Arial", 12))

input\_label.pack()

# 创建文本框及滚动条

text\_frame = tk.Frame(root)

text\_frame.pack(expand=True, fill="both")

global text # 声明text为全局变量

text = scrolledtext.ScrolledText(text\_frame, height=10, width=50, font=("Verdana", 10))

text.pack(expand=True, fill="both")

# 创建计算按钮

calculate\_button = tk.Button(root, text="计算FIRST集合", command=process\_input)

calculate\_button.pack()

# 创建结果显示框及滚动条

result\_label = tk.Label(root, text="计算结果：", fg="red", font=("Arial", 12))

result\_label.pack()

result\_frame = tk.Frame(root)

result\_frame.pack(expand=True, fill="both")

global result # 声明result为全局变量

result = scrolledtext.ScrolledText(result\_frame, height=10, width=50, state="disabled", font=("Verdana", 10))

result.pack(expand=True, fill="both")

# 进入主循环，显示窗口

root.mainloop()

def calculate\_first\_in(productions):

# 初始化FIRST集合的字典

firsts = {} # 保存FIRST集

completed = {} # 保存已计算过的非终结符

for non\_terminal in productions:

firsts[non\_terminal] = set()

completed[non\_terminal] = False

def calculate\_first(non\_terminal):

if completed[non\_terminal]:

return firsts[non\_terminal]

visited = set() # 避免循环依赖或重复访问 A->Ba|Bc;B->A

stack = [non\_terminal]

while stack:

current = stack.pop()

if current in visited:

continue

visited.add(current)

for production in productions[current]:

if production: # 检查产生式是否为空

double\_char = production[0:3] # if,id

if double\_char in terminal\_e:

firsts[non\_terminal].add(double\_char)

else:

first\_char = production[0]

if (first\_char.islower() or first\_char == 'ε' or first\_char.isdigit() or

first\_char in terminal\_b or first\_char in terminal\_c):

firsts[non\_terminal].add(first\_char)

elif first\_char in productions:

stack.append(first\_char)

completed[non\_terminal] = True # 保存已算的非终结符

return firsts[non\_terminal]

for non\_terminal in productions:

calculate\_first(non\_terminal)

return firsts

# 处理用户输入并计算FIRST集合

def process\_input():

productions = {} # {key:左部非终结符，value:[]}

# 获取输入的文法产生式

input\_text = text.get("1.0", tk.END).strip().split("\n")

for line in input\_text:

if line.strip() == '':

continue

left, right = line.split('->')

left = left.strip()

if left not in productions:

productions[left] = []

productions[left].extend([prod.strip() for prod in right.split('|')])

# 计算FIRST集合

firsts = calculate\_first\_in(productions)

result\_text = "\n".join(

[f"FIRST({non\_terminal}) = {{ {', '.join(first)} }}" for non\_terminal, first in firsts.items()])

result.config(state="normal")

result.delete("1.0", tk.END)

result.insert(tk.END, result\_text)

result.config(state="disabled")

# 主程序入口

if \_\_name\_\_ == "\_\_main\_\_":

# 定义终结符集合

terminal\_b = ['+', '-', '\*', '/', '^', '%', '&', '=', '!'] # 运算符

terminal\_c = ['[', ']', '{', '}', '(', ')', ',', '.', '\\', '"', '\'', '#', '@']

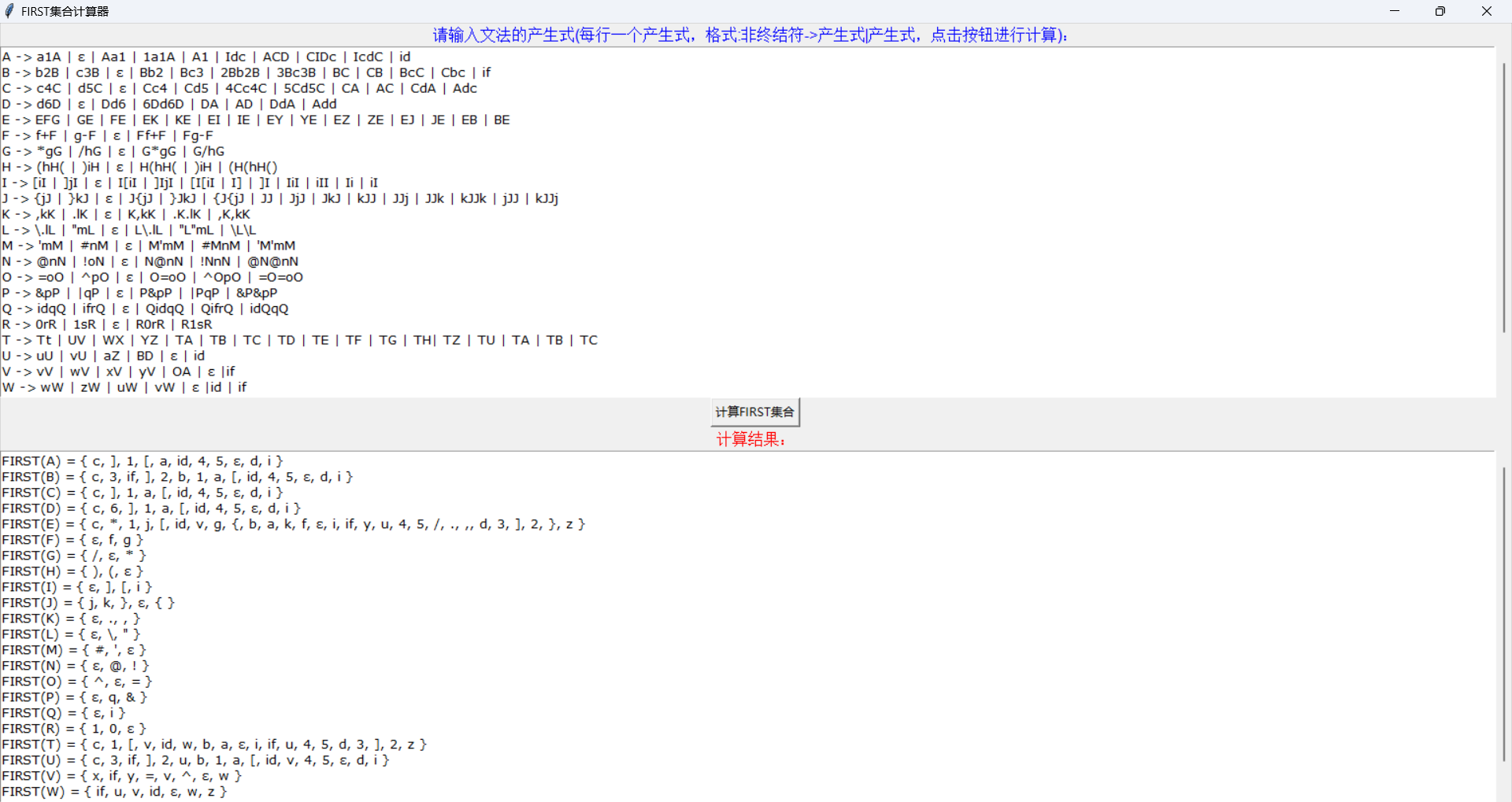
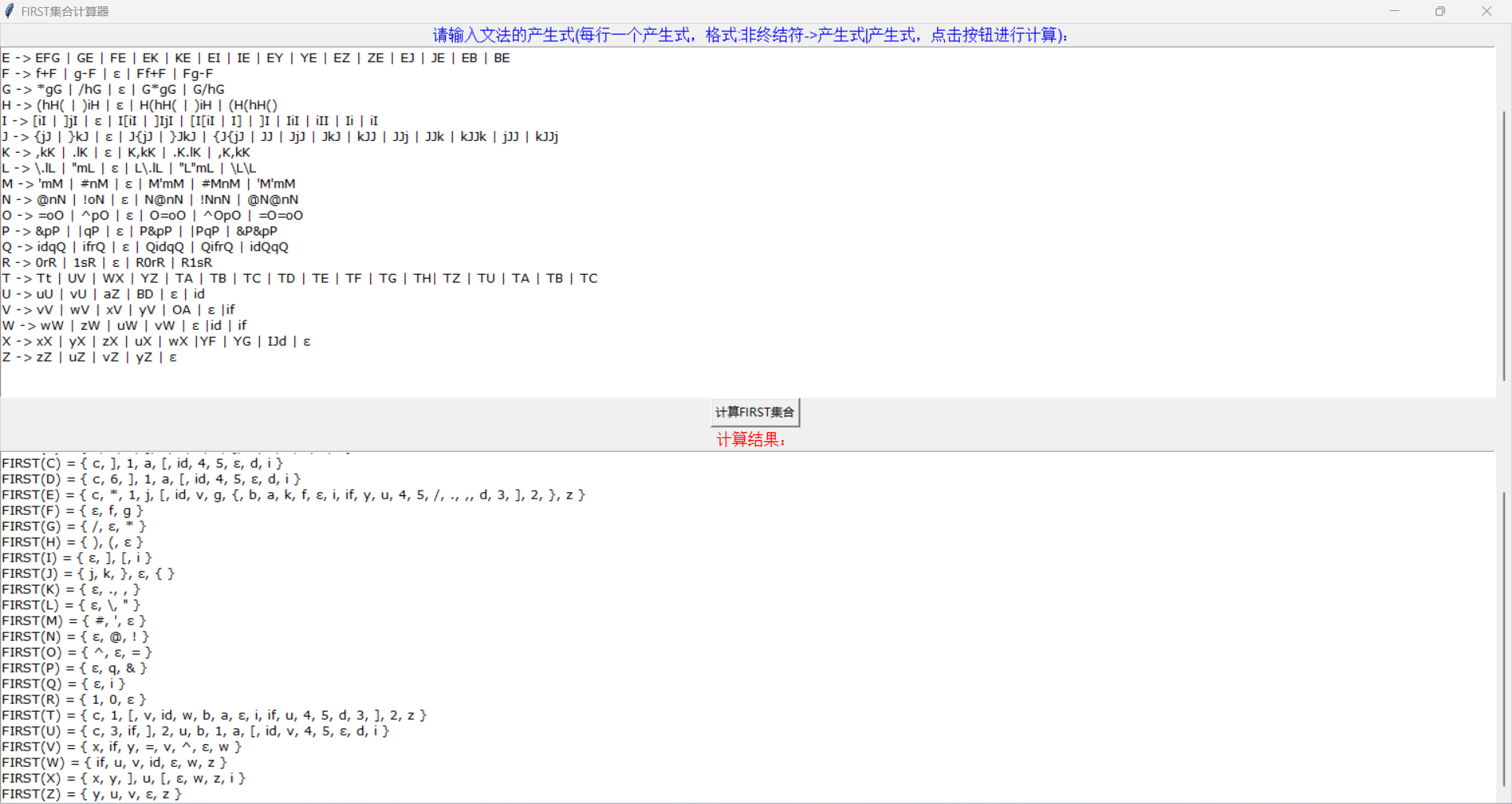
terminal\_e = ['id', 'if']

# 调用函数创建并显示窗口

create\_window()

**三、实验过程记录：**

实验结果截图：



**四、实验总结：**

本次实验旨在设计一个FIRST集合计算器，通过输入文法的产生式，计算出各非终结符的FIRST集合，并通过图形界面展示计算结果。主要步骤包括界面设计、算法实现和结果展示。

在界面设计方面，使用了Tkinter库创建了一个简单的图形界面，包括输入文本框、计算按钮和结果显示框。用户可以在文本框中输入文法产生式，点击按钮后，程序会计算FIRST集合，并将结果显示在结果框中。

算法实现部分主要涉及计算FIRST集合的逻辑。采用了基于栈的算法，通过遍历文法产生式，处理不同情况下的终结符、非终结符和空串，最终得到各非终结符的FIRST集合。

结果展示部分则负责将计算得到的FIRST集合以合适的格式展示在结果框中，方便用户查看和理解。

通过本次实验，加深了对文法分析中FIRST集合计算的理解，掌握了Tkinter库的基本使用和界面设计技巧。同时，也提升了对算法实现和结果展示的能力。