

学生学号	0122010870125	实验课成绩	
------	---------------	-------	--

# 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	计算机与人工智能学院
指导教师姓名	王云华
学 生 姓 名	雷裕庭
学生专业班级	软件 sy2001 班

2022 -- 2023 学年 第 一 学期

## 实验课程名称：编译原理

实验项目名称	设计并实现词法分析程序			实验成绩	
实验者	雷裕庭	专业班级	软件 sy2001	组别	NULL
同组者	NULL			实验日期	2022 年 12 月 1 日

### 第一部分：实验分析与设计（可加页）

#### 一、实验内容描述（问题域描述）

##### 【问题描述】

请根据给定的文法设计并实现词法分析程序，从源程序中识别出单词，记录其单词类别和单词值，输入输出及处理要求如下：

- （1）数据结构和与语法分析程序的接口请自行定义；类别码需按下表格式统一定义；
- （2）为了方便进行自动评测，输入的被编译源文件统一命名为 `testfile.txt`；输出的结果文件统一命名为 `output.txt`，结果文件中每行按如下方式组织：单词类别码 单词的字符/字符串形式(中间仅用一个空格间隔) 单词的类别码请统一按如下形式定义：

##### 【类别码标准】

单词名称	类别码	单词名称	类别码	单词名称	类别码	单词名称	类别码
标识符	IDENFR	if	IFTK	-	MINU	=	ASSIGN
整型常量	INTCON	else	ELSETK	*	MULT	;	SEMICN
字符常量	CHARCON	do	DOTK	/	DIV	,	COMMA
字符串	STRCON	while	WHILETK	<	LSS	(	LPARENT
const	CONSTTK	for	FORTK	<=	LEQ	)	RPARENT
int	INTTK	scanf	SCANFTK	>	GRE	[	LBRACK
char	CHARTK	printf	PRINTFTK	>=	GEQ	]	RBRACK
void	VOIDTK	return	RETURNTK	==	EQL	{	LBRACE
main	MAINTK	+	PLUS	!=	NEQ	}	RBRACE

#### 二、实验基本原理与设计（包括实验方案设计，实验手段的确定，试验步骤等，用硬件逻辑或者算法描述）

##### 【实验原理】

词法分析包括：用户自定义标识符、常数、字符串、关键字、界符的识别。用户自定义符号，顾名思义就是自己定义的变量名；函数名，常数包括整数、浮点数、科学计数；字符串包括 ‘ ’ 、 “ ” 两种形式的字符串；关键字就是程序内置的关键字，如 `int`、`main` 等；界符就是各类符号，如运算符、`{}`、`[]` 等。

词法分析的任务是，给定输入，识别输入序列中的单词，并将其正确分类。对于重复出现的单词，应该保证唯一性。

词法分析说容易也容易，人一眼就可以看出来一个“词”是属于哪一类的，然而让计算机来做就需要抽象出规则。常用的词法分析处理思路就是使用 DFA，这在常数识别上体现得淋漓尽致。

这个词法分析器是一个“总分结构”，各种识别功能在子程序写好，然后在主控程序

里调用即可。解释一下入口条件和出口条件：入口条件是主控程序的判断条件，指针读到的第一个字符决定主程序应该调用哪个子程序。出口条件是子程序执行期间的判断条件，决定何时单词识别结束、可以跳出。

**【输入形式】** testfile.txt 中的符合文法要求的测试程序。

**【输出形式】** 要求将词法分析结果输出至 output.txt 中。

**【样例输入】**

```
const int const1 = 1, const2 = -100;
const char const3 = '_';
    int change1;
    char change3;
int gets1(int var1,int var2){
    change1 = var1 + var2;
    return (change1);
}
void main(){
    printf("Hello World");
    printf(gets1(10, 20));
}
```

**【样例输出】**（部分）

```
CONSTTK const
INTTK int
IDENFR const1
ASSIGN =
INTCON 1
COMMA ,
IDENFR const2
ASSIGN =
MINU -
INTCON 100
SEMICN ;
CONSTTK const
CHARTK char
IDENFR const3
ASSIGN =
CHARCON _
SEMICN ;
INTTK int
IDENFR change1
```

三、主要仪器设备及耗材

PC 机

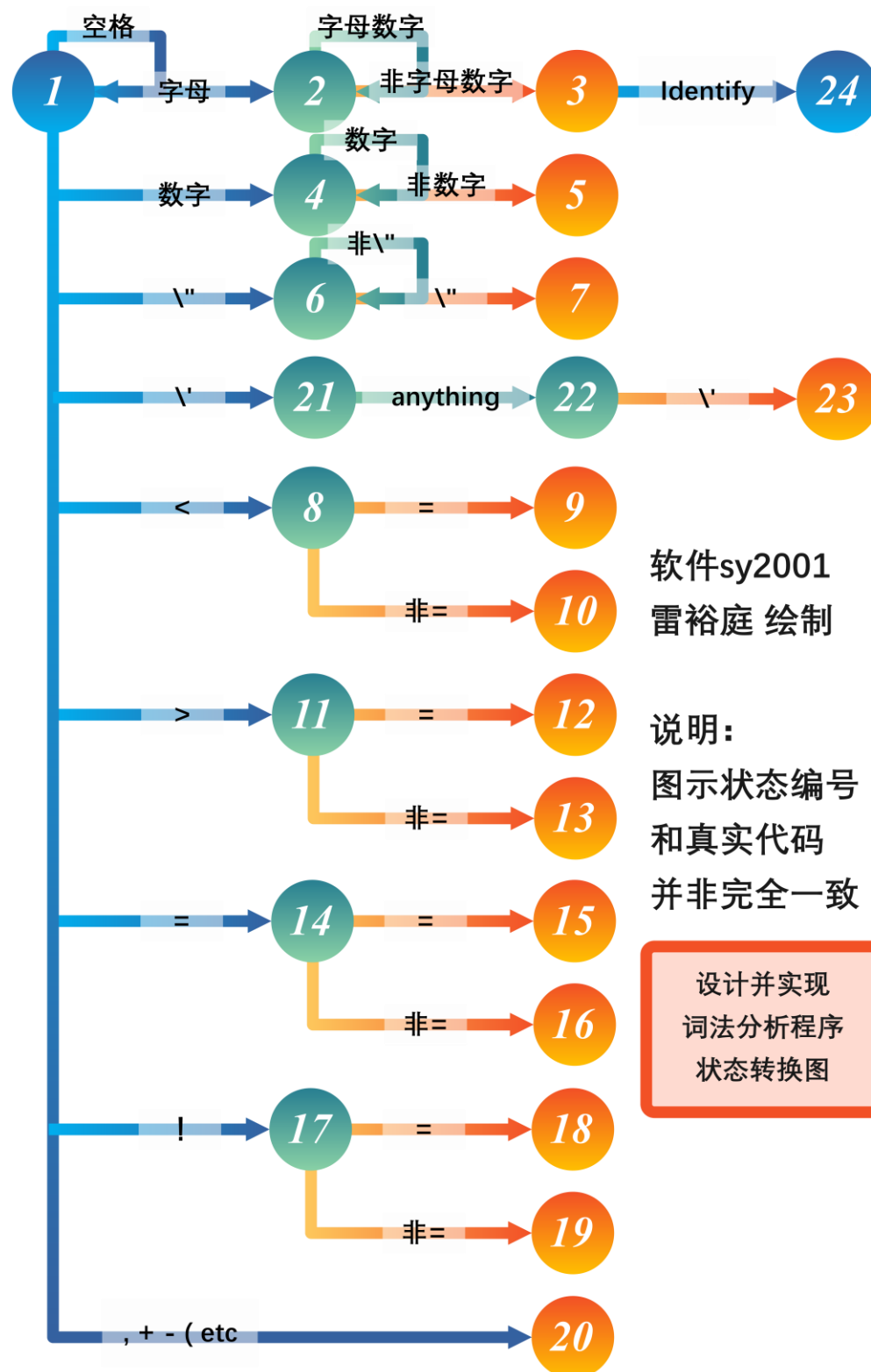
PyCharm 编译器

## 第二部分：实验调试与结果分析（可加页）

一、调试过程（包括调试方法描述、实验数据记录，实验现象记录，实验过程发现的问题等）

下面是主要的程序设计说明

1. 基于实验设计的状态转换图如下



## 2.主要的函数设计和调用

```
{'name': '标识符', 'class': 'IDENFR'},  
{'name': '字符常量', 'class': 'CHARCON'},  
{'name': '字符串', 'class': 'STRCON'},
```

### strClean:

该函数处理以下问题

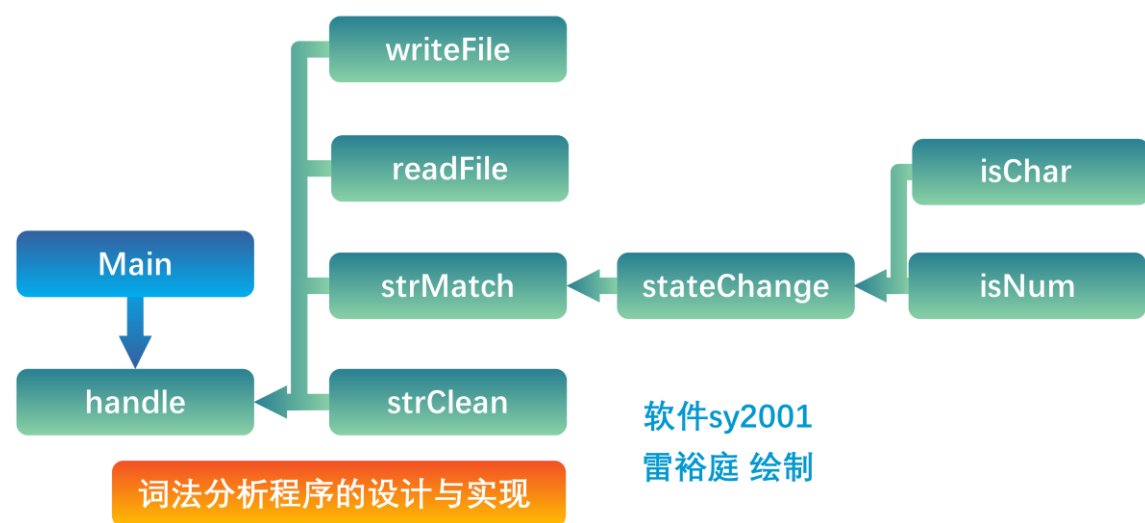
- 1.将 STRCON 从 CHARCON 中分出来
- 2.将 IDENFR 从 CHARCON 中分出来

### strMatch:

不断的读取字符串的字符，调用 **stateChange** 函数

### stateChange:

用于基于上一步状态和下一步的输入 char，判断下一步的状态和进行状态的储存



## 3. 主要实验代码

```
'''  
isChar 判断是否为字符  
'''  
  
def isChar(input):  
    input = str(input)  
    if ("Z" >= input >= "A") or ("z" >= input >= "a"):  
        return True  
    return False  
'''  
  
isNum 判断是否为数字  
'''  
  
def isNum(input):  
    input = str(input)  
    if "9" >= input >= "0":  
        return True  
    return False  
'''
```

stateChange 用于基于 输入状态 下一个字符的类型 返回下一个状态

'''

```
def stateChange(nowState: int, input: str, stateList: list, tempStr: str):
```

```
    # 多层处理
```

```
    print('nowState    input    tempStr')
```

```
    print(nowState, '          ', input, '          ', tempStr)
```

```
    if nowState == 1 and input == ":
```

```
        nextState = 1
```

```
    elif nowState == 1 and input == '':
```

```
        nextState = 1
```

```
    elif nowState == 1 and input == '\n':
```

```
        nextState = 1
```

```
    elif nowState == 1 and input is None:
```

```
        nextState = 1
```

```
    elif nowState == 1 and isChar(input):
```

```
        tempStr += input
```

```
        nextState = 2
```

```
    elif nowState == 1 and isNum(input):
```

```
        tempStr += input
```

```
        nextState = 4
```

```
    elif nowState == 1 and input == "<":
```

```
        nextState = 8
```

```
    elif nowState == 1 and input == ">":
```

```
        nextState = 11
```

```
    elif nowState == 1 and input == "!":
```

```
        nextState = 20
```

```
    elif nowState == 1 and input == "=":
```

```
        nextState = 21
```

```
    # 单一符号处理
```

```
    elif nowState == 1 and input == "-":
```

```
        stateList.append('MINU')
```

```
        stateList.append(input)
```

```
        nextState = 1
```

```
    elif nowState == 1 and input == "+":
```

```
        stateList.append('PLUS')
```

```
        stateList.append(input)
```

```
        nextState = 1
```

```
    elif nowState == 1 and input == "*":
```

```
        stateList.append('MULT')
```

```
        stateList.append(input)
```

```
        nextState = 1
```

```
    elif nowState == 1 and input == "/":
```

```
        stateList.append('DIV')
```

```
        stateList.append(input)
```

```

        nextState = 1
    elif nowState == 1 and input == ";":
        stateList.append('SEMICN')
        stateList.append(input)
        nextState = 1
    elif nowState == 1 and input == ",":
        stateList.append('COMMA')
        stateList.append(input)
        nextState = 1
    elif nowState == 1 and input == "(":
        stateList.append('LPARENT')
        stateList.append(input)
        nextState = 1
    elif nowState == 1 and input == ")":
        stateList.append('RPARENT')
        stateList.append(input)
        nextState = 1
    elif nowState == 1 and input == "[":
        stateList.append('LBRACK')
        stateList.append(input)
        nextState = 1
    elif nowState == 1 and input == "]":
        stateList.append('RBRACK')
        stateList.append(input)
        nextState = 1
    elif nowState == 1 and input == "{":
        stateList.append('LBRACK')
        stateList.append(input)
        nextState = 1
    elif nowState == 1 and input == "}":
        stateList.append('RBRACK')
        stateList.append(input)
        nextState = 1
    elif nowState == 1 and input == ":":
        stateList.append('COLON')
        stateList.append(input)
        nextState = 1
    # 第二层的计算 非终结
    elif nowState == 2 and (isChar(input) or isNum(input)):
        tempStr += input
        nextState = 2
    elif nowState == 4 and isNum(input):
        tempStr += input
        nextState = 4

```

```

elif nowState == 1 and input == "\":
    nextState = 6
elif nowState == 1 and input == "":
    nextState = 30
# 第二层计算终结
elif nowState == 2 and (isChar(input) == False and isNum(input) == False):
    stateList.append('IDENFR')
    stateList.append(tempStr)
    tempStr = ""
    nowState = 1
    nextState, tempStr = stateChange(nowState, input, stateList, tempStr)
elif nowState == 4 and isNum(input) == False:
    stateList.append('INTCON')
    stateList.append(int(tempStr))
    tempStr = ""
    nowState = 1
    nextState, tempStr = stateChange(nowState, input, stateList, tempStr)
'''
    "<":nextState=8
    ">":nextState=11
    "!=":nextState=20
    "=:nextState=21
    " :nextState=6
    ' :nextState=30
'''

elif nowState == 8 and input == "=":
    stateList.append('LEQ')
    stateList.append('<=')
    nextState = 1
elif nowState == 8 and input != "=":
    stateList.append('LSS')
    stateList.append('<')
    tempStr = ""
    nowState = 1
    nextState, tempStr = stateChange(nowState, input, stateList, tempStr)
elif nowState == 11 and input == "=":
    stateList.append('GTOE')
    stateList.append('>=')
    nextState = 1
elif nowState == 11 and input != "=":
    stateList.append('GRE')
    stateList.append('>')
    tempStr = ""
    nowState = 1

```



```

        nextState, tempStr = stateChange(nowState, input, stateList, tempStr)
elif nowState == 6 and input == "\":
    stateList.append('STRCON')
    stateList.append(tempStr)
    tempStr = ""
    nextState = 1
elif nowState == 6 and input != "\":
    tempStr += input
    nextState = 6
elif nowState == 30 and input == "":
    nextState = 1
elif nowState == 30 and input != "":
    stateList.append('CHARCON')
    stateList.append(input)
    nextState = 30
elif nowState == 21 and input == "=":
    stateList.append('EQL')
    stateList.append('==')
    nextState = 1
elif nowState == 21 and input != "=":
    stateList.append('ASSIGN')
    stateList.append('=')
    tempStr = ""
    nowState = 1
    nextState, tempStr = stateChange(nowState, input, stateList, tempStr)
elif nowState == 20 and input == "=":
    stateList.append('NEQ')
    stateList.append('!=')
    nextState = 1
elif nowState == 20 and input != "=":
    stateList.append('WRONG')
    stateList.append('!')
    tempStr = ""
    nowState = 1
    nextState, tempStr = stateChange(nowState, input, stateList, tempStr)
# 编译不能识别处理
else:
    print('unable to match')
    print(input)
    stateList.append('WRONG')
    stateList.append(input)
    nextState = 1
# 另一种处理方法      不处理
"""

```

```

else :
    print('unable to match')
    print(input)
    stateList.append(input)
    nextState=1
'''
return nextState, tempStr
'''
strMatch
'''
def strMatch(input: str):
    print('开始识别      按行识别')
    stateList = list()
    nowState = 1
    tempStr = ""
    for i in range(len(input)):
        inputC = input[i]
        nowState, tempStr = stateChange(nowState, str(inputC), stateList, tempStr)
    return stateList
'''
strClean 该函数处理以下问题
{'name': '标识符', 'class': 'IDENFR'},
{'name': '字符常量', 'class': 'IDENFR'},
{'name': '字符串', 'class': 'STRCON'},
1.将 STRCON 从 IDENFR 中分出来
2.将 IDENFR 从 IDENFR 中分出来
'''
def strClean(input: list):
    print('清理 IDENFR')
    input.append("")
    input.append("")
    for i in range(len(input) - 2):
        if input[i] == 'IDENFR':
            print(input[i + 1])
            if input[i + 1] == 'const':
                input[i] = 'CONSTTK'
                i -= 1
            if input[i + 1] == 'int':
                input[i] = 'INTTK'
                i -= 1
            if input[i + 1] == 'char':
                input[i] = 'CHARTK'
                i -= 1
            if input[i + 1] == 'void':

```

```

        input[i] = 'VOIDTK'
        i -= 1
    if input[i + 1] == 'main':
        input[i] = 'MAINTK'
        i -= 1
    if input[i + 1] == 'if':
        input[i] = 'IFTK'
        i -= 1
    if input[i + 1] == 'else':
        input[i] = 'ELSETK'
        i -= 1
    if input[i + 1] == 'do':
        input[i] = 'DOTK'
        i -= 1
    if input[i + 1] == 'while':
        input[i] = 'WHILETK'
        i -= 1
    if input[i + 1] == 'for':
        input[i] = 'FORTK'
        i -= 1
    if input[i + 1] == 'scanf':
        input[i] = 'SCANFTK'
        i -= 1
    if input[i + 1] == 'printf':
        input[i] = 'PRINTFTK'
        i -= 1
    if input[i + 1] == 'return':
        input[i] = 'RETURNTK'
        i -= 1
    del input[len(input) - 1]
    del input[len(input) - 1]
    return input
'''
readFile 读取文件 以 str list 的形式输出
'''
def readFile(address=r"C:\Users\leilu\Desktop\testfile.txt"):
    print("读取文件")
    with open(address, encoding='utf-8') as file:
        data = file.readlines()
    file.close()
    '''
    for i in range(len(data)):
        print(data[i])
    '''

```

```

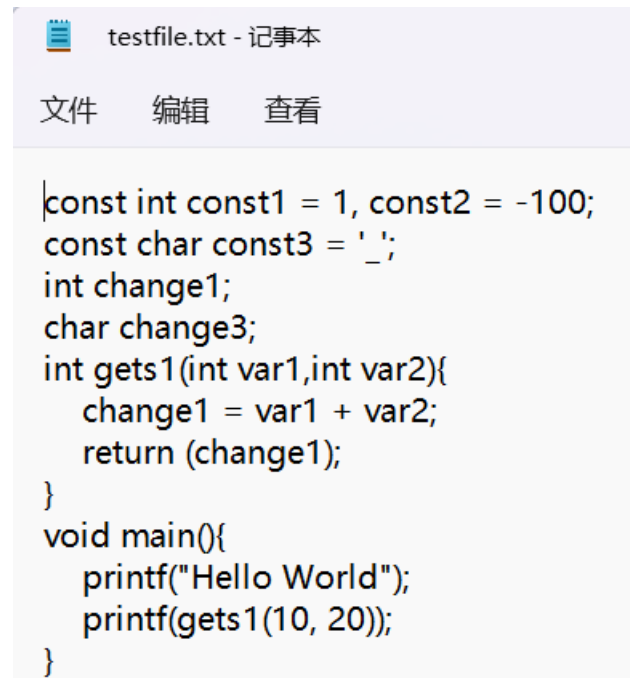
        return data
'''
writeFile 书写文件 以 str list 的形式输出
'''
def writeFile(data: list, address=r"C:\Users\leilu\Desktop\output.txt"):
    print("写入文件")
    with open(address, "w") as file:
        for i in range(len(data)):
            for j in range(len(data[i]) // 2):
                file.write(str(data[i][j * 2] + " " + data[i][j * 2 + 1]))
                file.write('\n')
            print(str(data[i][j * 2] + " " + data[i][j * 2 + 1]))
    file.close()
'''
handle 将以上的函数整合为可用
'''
def handle():
    data = readFile()
    output = list()
    for i in range(len(data)):
        tempStr = data[i]
        tempOut = strMatch(tempStr)
        tempOut = strClean(tempOut)
        tempSave = list()
        for j in range(len(tempOut)):
            tempSave.append(str(tempOut[j]))
        output.append(tempSave)
    writeFile(output)
def
                                compare(address1=r"C:\Users\leilu\Desktop\output.txt",
address2=r"C:\Users\leilu\Desktop\outExample.txt"):
    print("下面运行检验程序")
    aList = readFile(address1)
    bList = readFile(address2)
    '''
    #逐行检验
    print(len(aList),len(bList))
    for i in range(min(len(aList),len(bList))):
        print(aList[i],bList[i])
        print(aList[i]==bList[i])
    '''
    if (aList == bList):
        print("经过检验，编译结果准确无误")
    else:
        print("经过检验，编译结果有误")

```

```
    return aList == bList
if __name__ == '__main__':
    handle()
    compare()
```

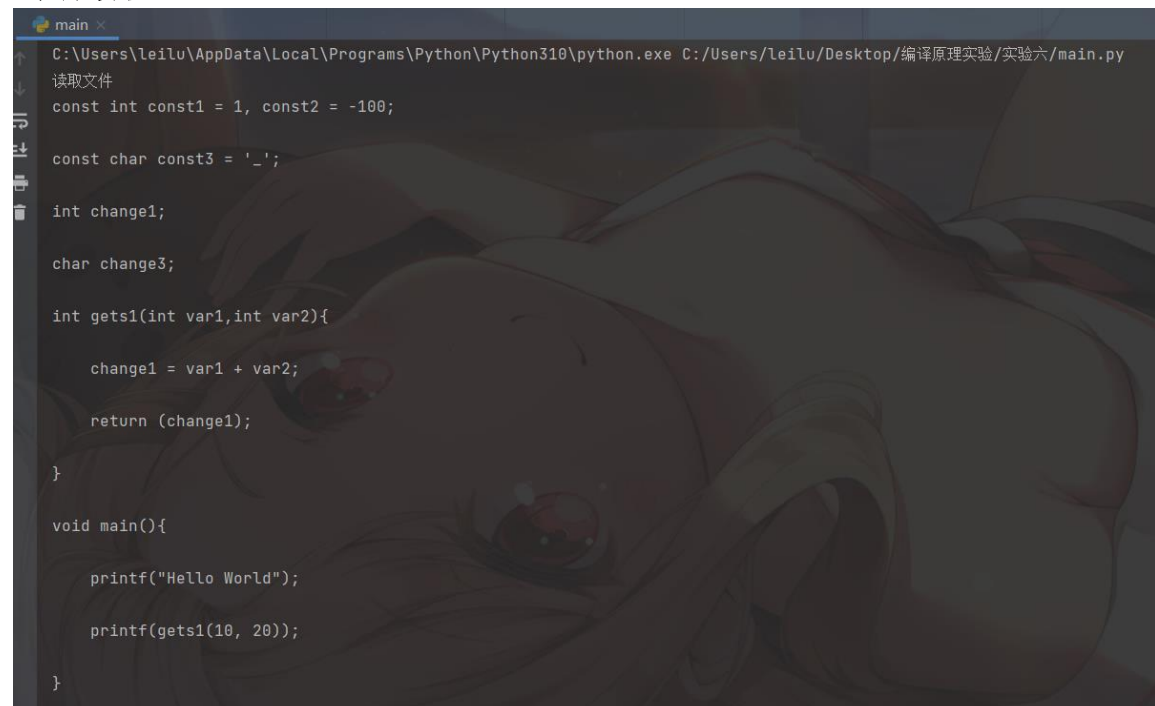
二、实验结果及分析（包括结果描述、实验现象分析、影响因素讨论、综合分析和结论等）

### 1.测试集



```
const int const1 = 1, const2 = -100;
const char const3 = '_';
int change1;
char change3;
int gets1(int var1,int var2){
    change1 = var1 + var2;
    return (change1);
}
void main(){
    printf("Hello World");
    printf(gets1(10, 20));
}
```

### 2.程序读取



```
C:\Users\leilu\AppData\Local\Programs\Python\Python310\python.exe C:/Users/leilu/Desktop/编译原理实验/实验六/main.py
读取文件
const int const1 = 1, const2 = -100;

const char const3 = '_';

int change1;

char change3;

int gets1(int var1,int var2){

    change1 = var1 + var2;

    return (change1);

}

void main(){

    printf("Hello World");

    printf(gets1(10, 20));

}
```

可见读取无异常

### 3.程序计算状态 和 储存字符串



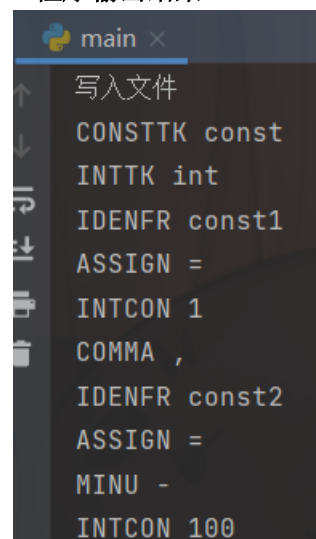
开始识别	按行识别	tempStr
nowState	input	tempStr
1	c	
nowState	input	tempStr
2	o	c
nowState	input	tempStr
2	n	co
nowState	input	tempStr
2	s	con
nowState	input	tempStr
2	t	cons
nowState	input	tempStr
2		const
nowState	input	tempStr
1		
nowState	input	tempStr
1	i	

### 4.从 CHARCON 中匹配标识符 STRCON



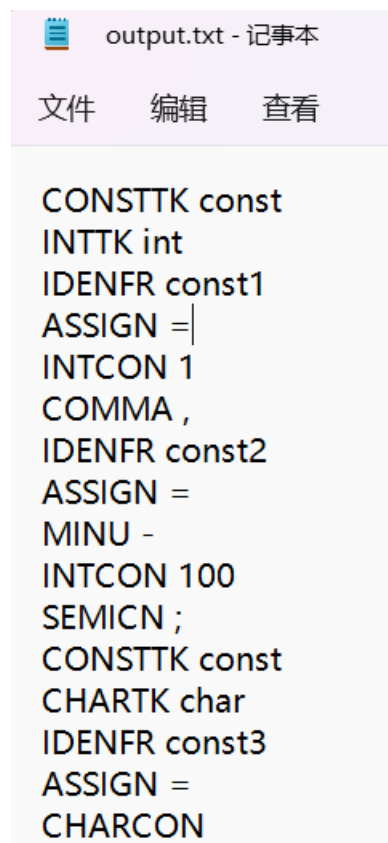
```
main x
清理IDENFR
const
int
const1
const2
```

### 5.程序输出结果



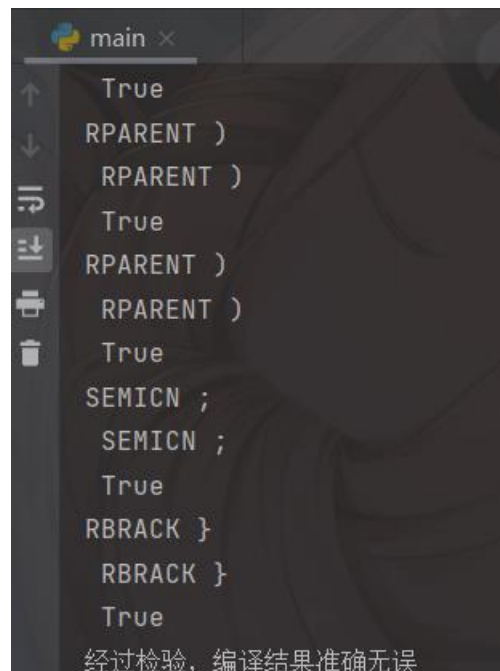
```
main x
写入文件
CONSTTK const
INTTK int
IDENFR const1
ASSIGN =
INTCON 1
COMMA ,
IDENFR const2
ASSIGN =
MINU -
INTCON 100
```

## 6.检测 TXT 写入情况



```
CONSTTK const
INTTK int
IDENFR const1
ASSIGN =|
INTCON 1
COMMA ,
IDENFR const2
ASSIGN =
MINU -
INTCON 100
SEMICN ;
CONSTTK const
CHARTK char
IDENFR const3
ASSIGN =
CHARCON _
```

## 7.程序判断是否和标答一致



```
True
RPARENT )
RPARENT )
True
RPARENT )
RPARENT )
True
SEMICN ;
SEMICN ;
True
RBRACK }
RBRACK }
True
经过检验，编译结果准确无误
```

## 8.代码展示

The image shows a Windows desktop environment. In the background, there is a large, semi-transparent image of a person in a white lab coat, possibly a doctor or scientist, with their hands clasped. In the foreground, a code editor is open, displaying two Python files side-by-side. The left file, named 'test.py', contains a function 'handle()' that reads a file, processes its content, and writes it back, along with a 'compare' function that compares two files. The right file, named 'test2.py', contains a 'stateList' and a 'stateChange' function that updates the list based on input. The Windows taskbar is visible at the bottom, showing the Start button, a search bar, and several application icons. The system tray in the bottom right corner shows the date and time as 2022/12/21 14:15.

### 三、实验小结、建议及体会

### 【实验小结】

通过了这次的实验，本人学会了解了词法分析程序的设计与实现，并且通过了实践将自己的所学进行的实践。

同时也进一步的熟练了 Python 的使用。

【建议】

还算是比较简单的实验，个人认为比之前的实验 13 要简单不少

### 【体会】

## 很简单的实验



