编号：_____

实习	一	二	三	四	五	六	七	八	九	十	总评	教师签名
成绩												

武汉大学计算机学院

《编译原理》课程

语法分析

实习报告

编　　号：CSA20

实习题目：构建一个小语言的词法分析程序

专业（班）：计科 10

学生学号：2019302060078

学生姓名：梁艺馨

任课教师：杜　卓　敏

２０２２　年 6　月 4　日

目录

1. 语言语法规则 2
2. 文法定义 3
3. 语法分析算法 3
3.1 语法分析功能 4
3.2 递归下降分析方法（递归子程序法） 4
3.3 递归下降语法分析部分关键函数 5
4. 出错处理出口 9
5. 测试计划 9
5.1 正确测试用例 9
5.2 错误测试用例 11

1. 语言语法规则

<程序> ::= <头文件定义>{<分程序>}
<头文件定义>::= #include<iostream> int main()
<分程序> ::= <变量定义><执行语句>
<变量定义>::= <变量类型><标识符>;{<变量定义 >}
<标识符> ::= <字母>{<字母>|<数字>}
<执行语句>::=<输入语句>|<输出语句>|<赋值语句>|<条件语句>|<While 语句>|<For 语句>|{<执行语句>}
<赋值语句>::=<标识符> <赋值运算符> <表达式 > ;
<While 语句>::= while(<条件语句>)do<执行语句>
<For 语句>::= for([<赋值语句>] <条件语句> ; <赋值语句>) <执行语句>
<条件语句>::= if(<条件>) then <执行语句> [else <执行语句>]
<逻辑与表达式>::= <逻辑或表达式> { || <逻辑或表达式> }
<逻辑或表达式>::= <逻辑与表达式> { && <逻辑与表达式> }
<关系运算>::= <表达式><关系运算符><表达式>
<表达式>::= <按位或表达式> { | < 按位或表达式>}
<按位或表达式>::= <按位与表达式> { & < 按位与表达式>}
<按位与表达式>::= <位移表达式> { <位移运算符> <位移表达式>}
<位移表达式>::= <乘除表达式>{<加减运算符> <乘除表达式>}
<乘除表达式>::= <运算式>{<乘除运算符> <运算式>}
<运算式>::=(<运算式>) |<标识符>|<整数>
<乘除运算符>::= * | /
<加减运算符>::= + | -
<位移运算符>::= >> | <<
<关系运算符> ::= == | != | < | <= | > | >= <赋值运算符>::= = | += | -= | *= | /=
<输入语句> ::= scanf(<标识符>)
<输出语句> ::= printf(<标识符>)
<变量类型>::= int | char | string
<字母> ::= a|b|...|X|Y|Z

<数字> ::= 0|1|2|...|8|9
<整数> ::= [-] <数字>

2. 文法定义

终结符

#include<iostream> int main() { } ; while do for [ ] ( ) if then else || && | & * / + - >> << == != < <= > >= = += -= *= /= scanf printf int char string

a...z A...Z 0...9

非终结符

<程序><头文件定义><分程序><头文件定义><变量定义><执行语句><变量类型><标识符><字母><数字><输入语句><输出语句><赋值语句><条件语句><While 语句><For 语句><执行语句><赋值运算符><表达式><逻辑或表达式><逻辑与表达式><条件><逻辑或表达式><关系运算><关系运算符><按位或表达式><按位与表达式><位移表达式><位移运算符><乘除表达式><加减运算符><乘除运算符> <运算式><标识符><整数><输入语句><输出语句><变量类型><整数>

部分非终结符命名	产生式
Program	<头文件定义>::= #include<iostream> int main()
Define	<变量定义>::= <变量类型><标识符>;{<变量定义 >}
IFCHECK	<条件语句>::= if(<条件>) then <执行语句> [else <执行语句>]
WhileCheck	<While 语句>::= while(<条件语句>)do<执行语句>
FORCHECK	<For 语句>::= for([<赋值语句>] <条件语句> ; <赋值语句>) <执行语句>
Printout	<输出语句> ::= printf(<标识符>)
Scanfin	<输入语句> ::= scanf(<标识符>)
Start	<执行语句>::=<输入语句>|<输出语句>|<赋值语句>|<条件语句>|<While 语句>|<For 语句>|{<执行语句>}
DoubleAnd	<逻辑或表达式>::= <逻辑与表达式> { && <逻辑与表达式> }
Relopcheck	<关系运算>::= <表达式><关系运算符><表达式>
OR	<按位或表达式>::= <按位与表达式> { & < 按位与表达式>}
ADDCheck	<位移表达式>::= <乘除表达式>{<加减运算符> <乘除表达式>}
MulCheck	<乘除表达式>::= <运算式>{<乘除运算符> <运算式>}
Equal	<赋值语句>::=<标识符> <赋值运算符> <表达式 > ;

3. 语法分析算法

3.1 语法分析功能

语法的分析是依据语法规则，逐一分析词法分析时得到的单词串，把单词串分解成各类语法单位，即确定它们是怎样组成说明和语句，以及说明和语句又是怎样组成程序的。分析时如发现有不合语法规则的地方,便将出错的位置及出错性质打印报告给程序员;如无语法错误，则用另一种中间形式给出正确的语法结构，供下一阶段分析使用。

3.2 递归下降分析方法（递归子程序法）

3.2.1 基本思想

对每一个语法成分（用非终结符号代表），构造相应的分析子程序，该分析子程序分析相应于该语法成分（非终结符号）的符号串。

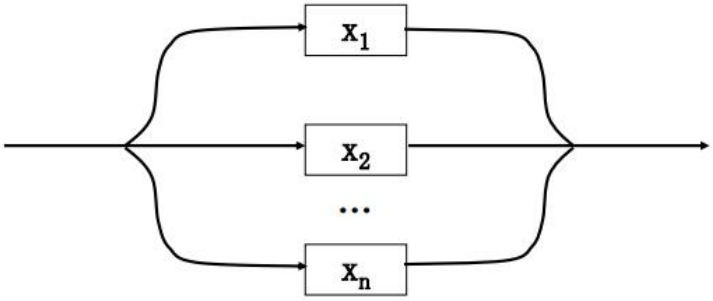
由于语法成分之间不可避免会含有递归，所以分析子程序之间也会有递归调用，故而又称为递归子程序法

3.2.2 分析过程

从开始符号出发，在语法规则支配下，逐个扫描输入符号串中的符号，根据文法和当前的输入符号预测到下一个语法成分是 U 时，便确定 U 为目标，并调用 U 的分析子程序 P(U)工作。在 P(U)工作的过程中，又有可能确定 U 或其它非终结符号为子目标，并调用相应的分析子程序。如此继续下去，直到得到结果。

3.2.3 分析子程序构造方法

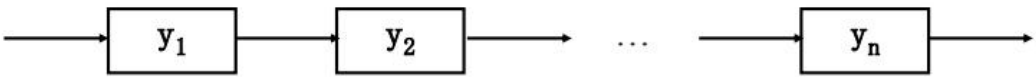
1）对于每个非终结符号 U，编写一个相应的子程序 P(U)。



2）对于产生式 U→x1 |x2 |…|xn，有一个关于 U 的子程序 P(U) 。U 可空

```
IF CH IN FIRST(x1 ) THEN P(x1 )
    ELSE IF CH IN FIRST(x2 ) THEN P(x2 )
        ELSE …
        …
        IF CH IN FIRST(xn ) THEN P(xn )
        ELSE IF not(CH IN FOLLOW(U)) THEN ERROR
```

3）对于 x=y1y2…yn； BEGIN P(y1)；P(y2)；…；P(yn) END 。

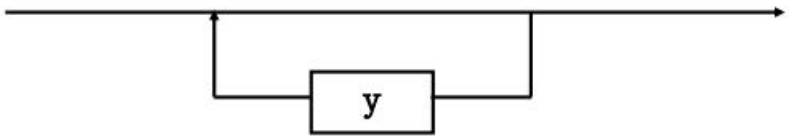


如果①yi∈VN，则 P(yi)就代表调用处理 yi 的子程序；

②yi∈VT，则 P(yi)为形如下述语句的一段程序

IF CH=yi THEN READ(CH) ELSE ERROR

4）如果 x={y}，在程序中就是一个循环。



3.3 递归下降语法分析部分关键函数

部分非终结符转移，如<字母><数字><整数><加减运算符>等直接在函数中进行实现

1. 检测头文件

<程序> ::= <头文件定义>{<分程序>}

<头文件定义>::= #include<iostream> int main()

```
void GrammerAnalyze::Program()

{...Define();    ...Start();...}
```

2. 变量定义

<变量定义>::= <变量类型><标识符>;{<变量定义 >}

```
void GrammerAnalyze::Define()

{...Define()...}
```

3. 变量赋值

<赋值语句>::=<标识符> <赋值运算符> <表达式 > ;

```
void GrammerAnalyze::Numgiven(){

if (...)//检查是否被定义

}
```

4. If 语句

<条件语句>::= if(<条件>) then <执行语句> [else <执行语句>]

```
void GrammerAnalyze::IFCHECK(){

...Bool()...
```

...Start();...

}

5. while 语句

<While 语句>::= while(<条件语句>)do<执行语句>

```
void GrammerAnalyze::WhileCheck(){

...Bool()...

...Start();...

}
```

6. for 语句

<For 语句>::= for([<赋值语句>] <条件语句> ; <赋值语句>) <执行语句

```
void GrammerAnalyze::FORCHECK(){

...Numgiven();...

...Bool();...

...Start();...

}
```

7. 输出语句

<输出语句> ::= printf(<标识符>)

```
void GrammerAnalyze::Printout(){

...OR();...

}
```

8. 输入语句

<输入语句> ::= scanf(<标识符>)

```
void GrammerAnalyze::Scanfin(){

...OR()...

}
```

9. 执行语句

<执行语句>::=<输入语句>|<输出语句>|<赋值语句>|<条件语句>|<While 语句>|<For 语句>|{<执行

语句>}

```
void GrammerAnalyze::Start()

{

    ...IFCHECK(); ...

     ...Start();...

    ...WhileCheck();...

     ...Start();...

     ... Numgiven();

   ...FORCHECK(); ...

      ...Printout();...

      ...Scanfin();...

}
```

10. ||判断

<逻辑与表达式>::=<逻辑或表达式>{ || <逻辑或表达式> }

```
string GrammerAnalyze::Bool(){

...DoubleAnd();...

}
```

11. &&判断

<逻辑或表达式>::=<逻辑与表达式>{ && <逻辑与表达式> }

```
string GrammerAnalyze::DoubleAnd() {

...Relopcheck();...

}
```

12. 关系运算 >= > <= < != ==

<关系运算>::=<表达式><关系运算符><表达式>

```
string GrammerAnalyze::Relopcheck(){

...OR();...
```

}

13. OR 判断

<按位或表达式>::=<按位与表达式>{ & < 按位与表达式>}

string GrammerAnalyze::OR(){

...AND()...

}

14. AND 判断

<按位与表达式>::=<位移表达式>{<位移运算符><位移表达式>}

string GrammerAnalyze::AND(){

...Drift();...}

15. 移位

<按位与表达式>::=<位移表达式>{<位移运算符><位移表达式>}

string GrammerAnalyze::Drift(){

...ADDCheck();...

}

16. 相加判断

<位移表达式>::=<乘除表达式>{<加减运算符><乘除表达式>}

string GrammerAnalyze::ADDCheck(){

...MulCheck();...

}

17. 乘法判断

<乘除表达式>::=<运算式>{<乘除运算符><运算式>}

string GrammerAnalyze::MulCheck(){

...Equal();...

}

18. 判断语句是否合法

```
string GrammerAnalyze::Equal(){

...OR();...

}
```

4. 出错处理出口

当在函数内检测到错误时，输出错误类型并退出，错误类型定义：

1. 尝试打开错误的文件
2. 使用了不合法的符号
3. 头文件未声明或声明错误
4. 未进行变量定义
5. 错误定义变量
6. 条件语句作为执行
7. 赋值语句使用错误
8. While 语句没有配套的 do
9. While 语句之间定义了变量
10. While 语句之后缺少判断语句
11. While 语句组成不完整
12. for 语句缺少组成成分
13. 在 for 语句之中定义变量
14. 重定义相同的变量
15. 在 While 语句之中定义了变量
16. if 语句组成不完整
17. if 语句没有配套的 then
18. then 没有后续语句
19. {没有配套的}作为结尾
20. 赋值表达式没有;作为结尾
21. 变量定义语句没有;作为结尾
22. 输入语句没有正确书写
23. 输出语句没有正确书写
24. 输入语句输入了多个变量或表达式
25. 输出语句输出了多个变量或表达式
26. （没有）作为匹配
27. 文法没有执行语句
28. 文法没有定义语句
29. 使用了未定义的变量
30. 除法表达式使用 0 作为分母

5. 测试计划

5.1 正确测试用例

```
#include <iostream>
```

```
int main()
{
    int a; int b2b; int i ; int j;int b;
    char c;
    string str;
    scanf(a)
    scanf(c)
    scanf(str)
    printf(a)
    printf(c)
    printf(str)
    for(i = -3 ; i<0 ; i +=1)
    {
        j = 1 ;
        for( ;  j <= 1  &&  j != 2   ;  j +=  2)
        {
              scanf(b)
              while(b2b != 0 && a != 0)
              do{
                  a = (a/a) - (a+a) * (a-a) - a ;
                  b2b=  b2b <<  1 ;
        }
```

```
                printf(a)
                  a = a|1;
        }
        printf(b2b)
    }
    printf(a)
    printf(b)
    printf(c)
    b = b& 1;
    b = 2;
    while(b != 0)
    do{
         scanf(a)
         if(a != 0)
         then
        {
    if(a == c || a>= b ||  a< b ||  (a+b)> ( b - 4 ) /b * 3   && a != b)
     then a += 1 * 2 - b/1 ;
     else a -= ( 1 + 1)  - ( b * 1 );

    a *= 0;
    b /= 2;
        }
        else
        {
                for(i = 1 ; i<= 4 ; i+=1)
           a += ( a  * b )  - ( b >> 2 );
    a -=  (a + 1 )   << 1;
    printf(a)
         }
         b = b - 1 ;
    }
    printf(a)
    printf(b)
    printf(c)
    printf(str)
}
```

5.2 错误测试用例

分几个测试案例进行，在此处测试了 12 种错误

1.

```
#include <iostream>
int main()
{
```

%
}

```
不合法字符     %    位置：4行1列
```

2.

```
#include<cstdio>
int main()
{
}
```

```
Expect for 'iostream' to start
```

3.

```
#include<iostream>
int main()
{
      int a;
    a = 3
}
```

```
缺少';'      错误位置:6行1列
```

4.

```
#include<iostream>
int main()
{
      int a;
a = 3;
```

```
缺少}作为结尾位置 5行9列
```

5.

```
#include<iostream>
int main()
{
      int a;
      while a == 3)
}
```

```
缺少(      错误位置:5行10列
```

6.

```
#include<iostream>
int main()
{int a;
if(a = 3)
a = 4;
}
```

7.

```
#include<iostream>
int main()
{
    int a;
    while(a == 1) a-=1;
}
```

8.

```
#include<iostream>
int main()
{
      int a;int b;
      scanf(a b)
}
```

9.

```
#include<iostream>
int main()
{
      int a; int b;
      a = a+ ;
}
```

10.

```
#include<iostream>
int main()
{
int b;
      a = 3;
}
```

11.

```
#include<iostream>
int main()
{
      int a;int a;
}
```

12.

```
#include<iostream>
int main()
{
	int a;
	for(int b = 3 ; b < 4 ; b+= 1)
}
```