实验七 对算术表达式构造递归下降翻译器

一、实验内容

【任务性质】必做任务，分值10分。

【任务介绍】对算术表达式做递归下降分析，同时将其翻译为中间代码。

【输入】算术表达式。

【输出】四元式序列。

【题目】对实验四的程序进行升级改造，使得程序对于输入的任意一个算术表达式，在对其做递归下降分析的同时，生成等价的中间代码，一遍完成。要求：

```
<E> -> <T><E1>
<E1> -> <A><T><E1> | empty
<T> -> <F><T1>
<T1> -> <M><F><T1> | empty
<F> -> id | number | (<E>)
<A> -> + | -
<M> -> * | /
```

1. 基础文法：同实验四。
2. 语法分析：沿用实验四的程序框架。
3. 语义处理：生成四元式序列。
4. 一遍处理：把语义处理的代码插入到语法分析的代码中。设计要点有两个：
    1. 语法分析走到哪里，应该执行相应的语义动作；
    2. 语义动作（这里指的是生成四元式）应该怎么做。
5. 为简化问题，不考虑输入有错误的情况，不考虑语义检查。

二、实验思路

1. 四元式

形如下图

例 6.6 赋值语句 a = b * -c + b * -c 的三地址代码如图 6-10a 所示。这里我们使用特殊的 minus 运算符来表示“-c”中的单目减运算符“-”，以区别于“b-c”中的双目减运算符“-”。请注意，单目减的三地址语句中只有两个地址，复制语句 $a = t_5$ 也是如此。

图 6-10b 描述了实现图 6-10a 中三地址代码的四元式序列。 □

```
t1 = minus c
t2 = b * t1
t3 = minus c
t4 = b * t3
t5 = t2 + t4
 a = t5
```

a) 三地址代码

	op	arg_1	arg_2	result
0	minus	c		t_1
1	*	b	t_1	t_2
2	minus	c		t_3
3	*	b	t_3	t_4
4	+	t_2	t_4	t_5
5	=	t_5		a
	...			

b) 四元式

- 以结构体quad存储四元式

```
struct quad {
    string op, arg1, arg2, res;
};
//op    双目操作符
//arg1  第一个操作数
//arg2  第二个操作数
//res   运算结果
```

2. 生成四元式

- **与中序转后序类似（见实验四实验报告）**，构建两个栈：操作数栈和运算符栈
- 与实验四处理相同，递归下降分析算术表达式，**分析到F, A, M产生式时，会遇到终结符，此时添加语义动作**如下：
  - （：**F产生式**中，加入**语义动作**——入栈（入栈后左括号优先级降为最低，确保其他符号正确入栈）
- ）：**F产生式**中，加入**语义动作**——)意味一组括号结束，不断弹出运算符栈顶和操作数栈两个数构成四元式，直到遇到左括号，左括号仅弹出即可
  - 如果是操作数(id, number)，即**F产生式**，加入**语义动作**——进入操作数栈
  - 如果是运算符(+, -, *, /)（方便描述，命名为token），即**A和M产生式**，加入如下**语义动作**
    - 如果运算符栈不空，或token优先级>栈顶，则入栈
    - 如果运算符栈不空，或token优先级<=栈顶，则出栈栈顶运算符和出栈操作数栈两个操作数用来生成四元式，直到运算符栈空，或者token优先级>栈顶，再将token入栈
  - 若算术表达式分析完成，运算符栈仍有剩余，则依次出栈生成四元式
    - (ps: !!!!!!bug，一个边界情况)

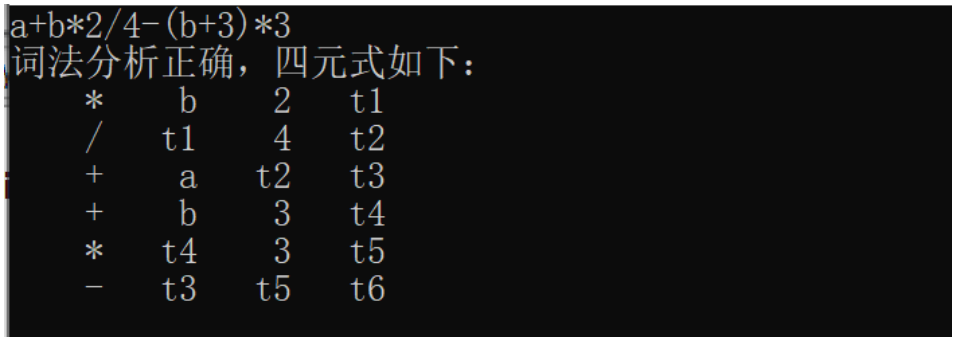
      当算术表达式是由一个token组成的时候，操作数栈只有一个元素，而运算符栈为空，所以这种情况下也要考虑到，直接输出(_, id/number, _, _)
    - 实际上，操作数栈最后总会有一个元素，即整个表达式的结果
- 上述生成四元式的过程中，result用ti表示，同时将ti替换出栈的两个操作数，进入操作数栈

三、实验环境和结果

- 语言：C++
- 编译环境：Code Blocks 17.12 自带MinGW中的g++，需要在编译器设置中勾选C++14
- 输入1：

```
a+b*2/4-(b+3)*3
```

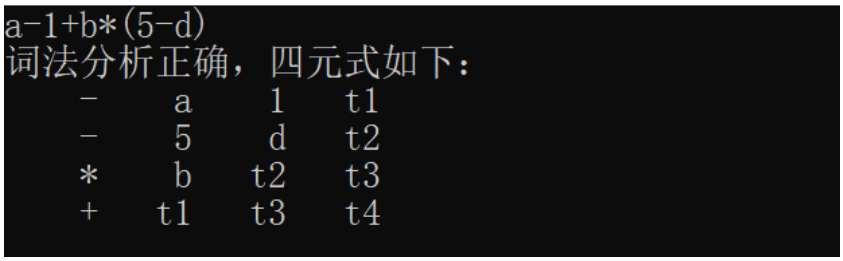
输出1：



- 输入2：

```
a-1+b*(5-d)
```

输出2：



- 输入3（边界）：

```
0
```

输出3：