说明：

本次程序根据递归下降法实现了给定如下文法的语法分析树的字符串

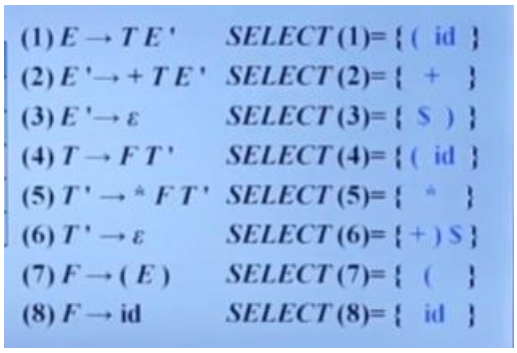
E → TE'

E' → +TE' | ε

T → F T'

T' → * F T' | ε

F → (E) | id



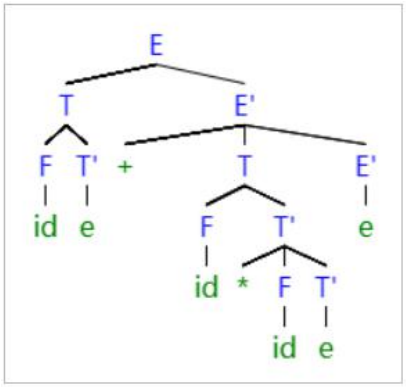
本程序需要在运行后的控制界面输入需要生成语法分析树的字符串，字符串每个字符都要以空格隔开，并且由于文法限制仅能识别由（id，+，*，(,)）构成的字符串的表达式。如果输入不符合文法，则会报错，输出 Input error!

程序根据表达式分别定义了五个函数方法，以对应非终结符名命名（E’则命名为E1,以此类推），函数返回值为结果字符串和下一个字符，函数思路较为简单，对应每个表达式的右端遍历，如果为终结符则拼接返回，如果为非终结符则调用对应非终结符的函数，直至输入字符串全部遍历完，递归结束。还创建了字符读取函数 getnext（）和结果字符拼接函数 add_res（）来辅助完成。

运行结果：

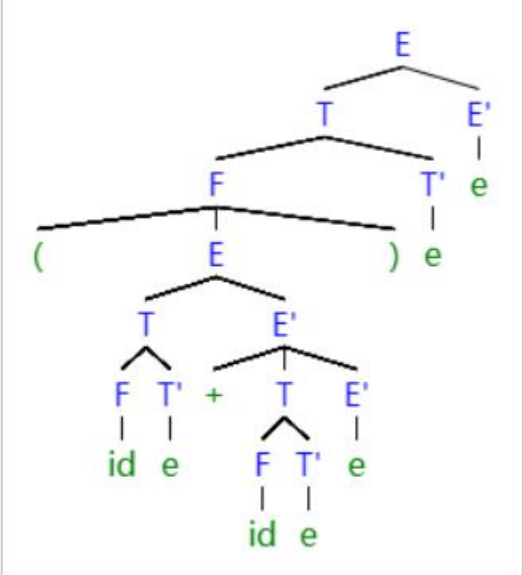
输入：id + id * id

输 出 ： [E[T[F[id]] [T'[e]]] [E'[+] [T[F[id]] [T'[*] [F[id]] [T'[e]]]] [E'[e]]]]



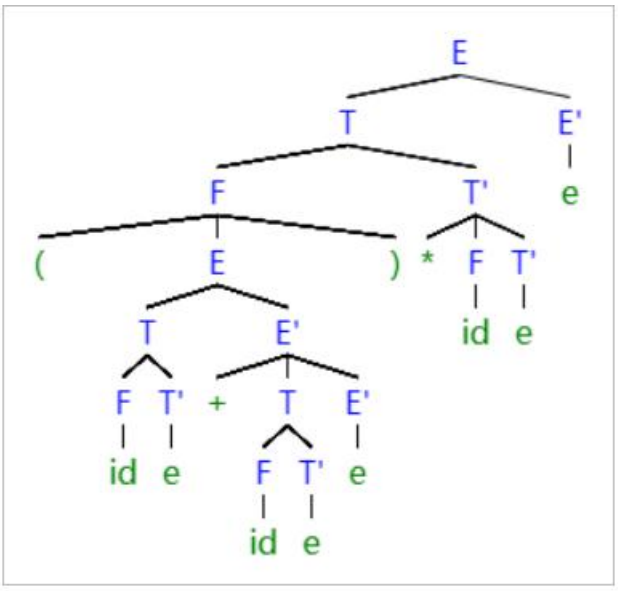
输入：(id + id)

输出： [E[T[F[(] [E[T[F[id]] [T'[e]]] [E'[+] [T[F[id]] [T'[e]]] [E'[e]]]] [)]] [T'[e]]] [E'[e]]]



输入：(id + id) * id

输出： [E[T[F[(] [E[T[F[id]] [T'[e]]] [E'[+] [T[F[id]] [T'[e]]] [E'[e]]]] [)]] [T'[*] [F[id]] [T'[e]]]] [E'[e]]]



输入：(id * id) + (id + id) * (id + id)

输出： [E[T[F[(] [E[T[F[id]] [T'[*] [F[id]] [T'[e]]]] [E'[e]]] [)]] [T'[e]]] [E'[+] [T[F[(] [E[T[F[id]] [T'[e]]] [E'[+] [T[F[id]] [T'[e]]] [E'[e]]]] [)]] [T'[*] [F[(] [E[T[F[id]] [T'[e]]] [E'[+] [T[F[id]] [T'[e]]] [E'[e]]]] [)]] [T'[e]]]] [E'[e]]]]