当token流并无法满足我们的rule的时候，就会生成syntax error

当前输入的next token没有满足期待，就会有syntax error

compiler的一个主要作用就是识别identify所有的programming error，并且给出有意义的指示，关于在Input的那个位置以及nature//性质种类

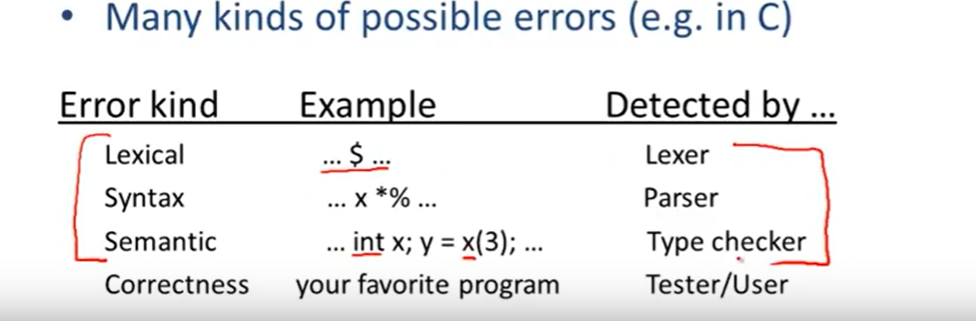
5个要点  
detect所有compile-time error

report所有错误的存在，清除准确

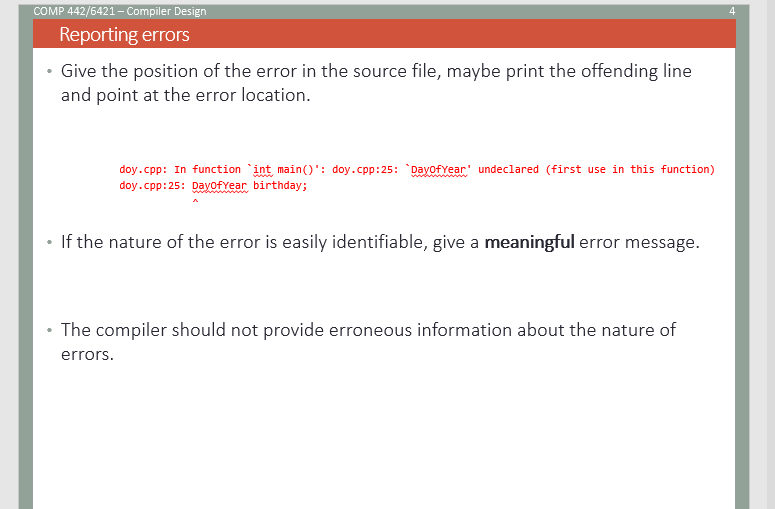
recover，尽快修复每个error这样能检测到剩下的error

不应该减慢正确program的处理

避免虚假的错误spurious errors，只是一串提早的error

parser需要纠正的是syntax error

Error reporting.



给出error在原有source file的位置，应该Print不愉快的Line并且指出error位置

如果error的性质能够辨别，应该给出一个有意义的error信息

compiler不应该提供错误的信息

当遇到一个syntax error的时候,parser应该能够继续处理分析的过程

一个好的error recovery应该尽可能提早检测到error

bottom-up parser能更快的检测到error

error recover也应该尽快这样就不会漏过其他的subsequent error。 优秀的error recovery应该尽可能跳过比较少的token

不要产生超过实际存在的error

如果没有遇到error不要过快处理导致超负

应该避免report由error recovery产生的其他错误

panic mode：

当发现一个error的时候，parser直接抛弃之后所有的input token一直到遇到指定的synchronizing tokens。通常是分隔符例如分号或者}

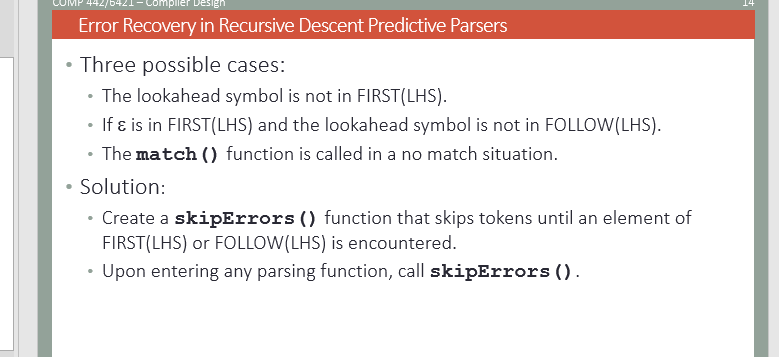
常用的方法是把synchronizing tokens形成一组FIRST |FOLLOW集合

负面效果：可能会跳过其他的error

Error Recovery in recursive descent predictive parsers

我们具体的情况

三种可能：



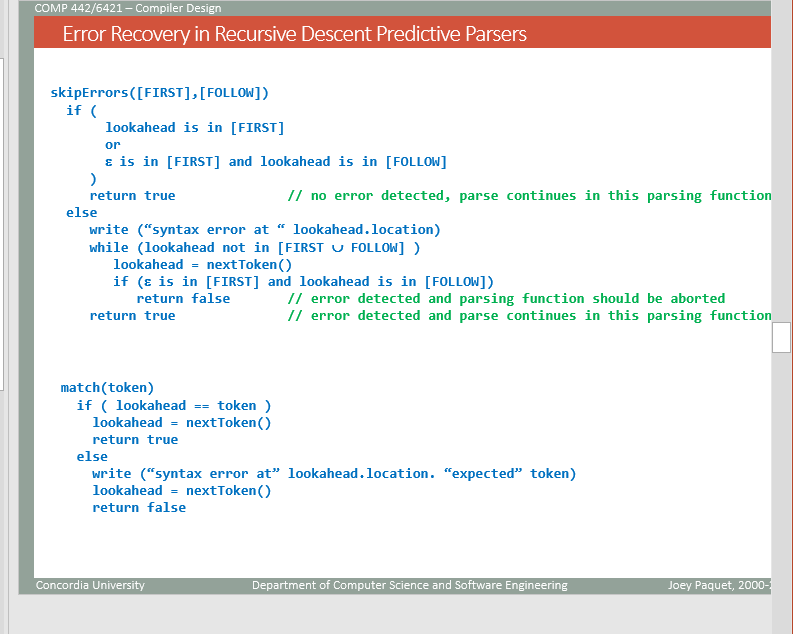
lookahead(当前input符号)不在first里

如果存在A->e形式，然后不在FOLLOW里

match() function没有找到对应的match情况

解决方案：

创造一个skipErrors() function， 一直skip token直到遇到first 或follow 中的其中一个elemen



skip errors， 如果在FIRST或者存在e然后在FOLLOW里，return true，代表没有error

不然print有syntax error

如果不在这两个集合里，一直next token

如果这时遇到的新input在follow里且e里有first， 直接终止整个程序

如果在FIRST 里|在follow里且e不再first里

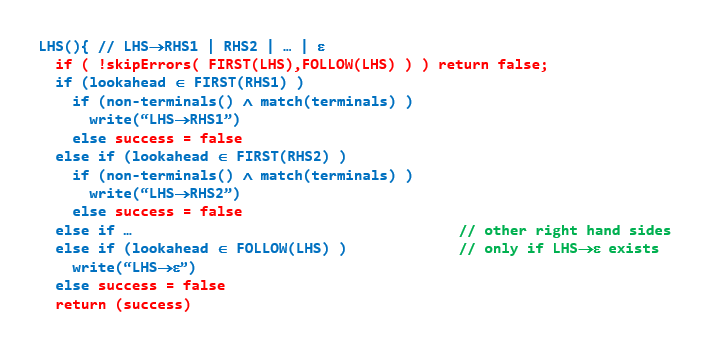
继续，成功报错

match的报错

如果=token成功

不然就报错，找到下一个token

returnfalse

写法改变，如果！skiperror这些集合， 也就是说成功报错并且找到token，不会return false，如果是e在first里又遇到follow，会return false

如果成功,write

不然 success=false

最后return success

