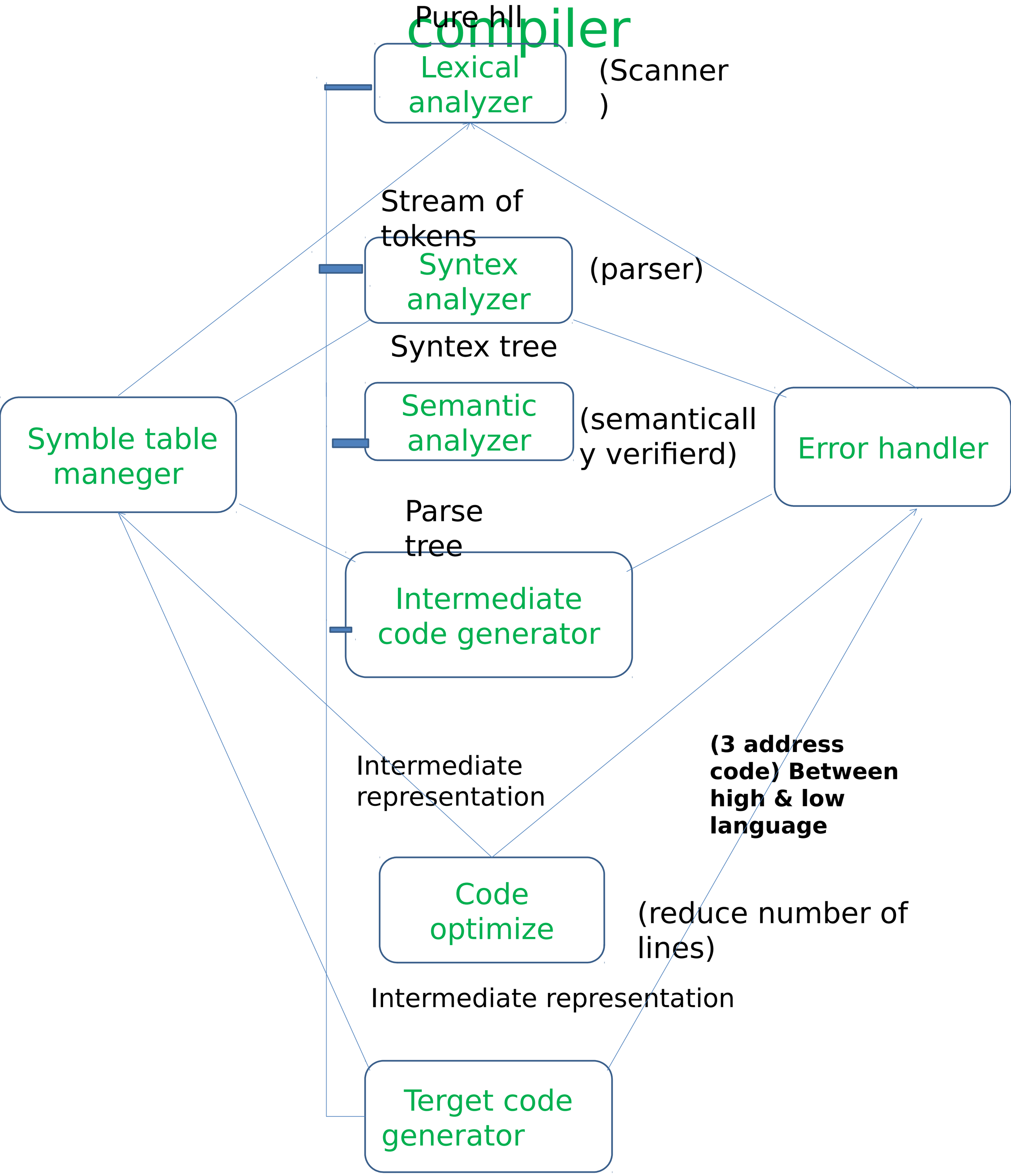
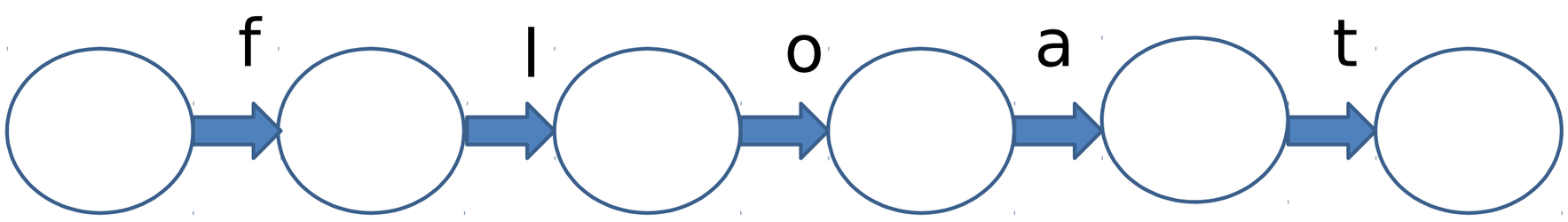
Computer Deisgn

The structure of a compiler



Lixeme,Pattern,Token

Token
<Token name,aH value>
positio
n



Symble table manager

s/n	v/n	v/t
01	Positio n	float
02	initial	float
03	Rate	float

float | positio n | initial | rate | ; | /*correct*/

Position = initial+ rate*60

Lexical analyzer | L(l+d)*

<id,1><=><id,2><+><id,3><*><60>
id,1 = id2 + id3 *60

A lexeme is a sequence of character matching the pattern

Token	Sample lexeme	pattern
it	it	it
Relation	<,<=,=,<>,>,>=	<or<=or=or<>or >or>=

Compiler Deisgn

Intoduction to lexecal analyzer

/*find sum of x and y*/

Int sum (x,y)
Return (x+y)

Compiler Deisgn

Introduction to syntex analyzer

Float,position,initial,rate

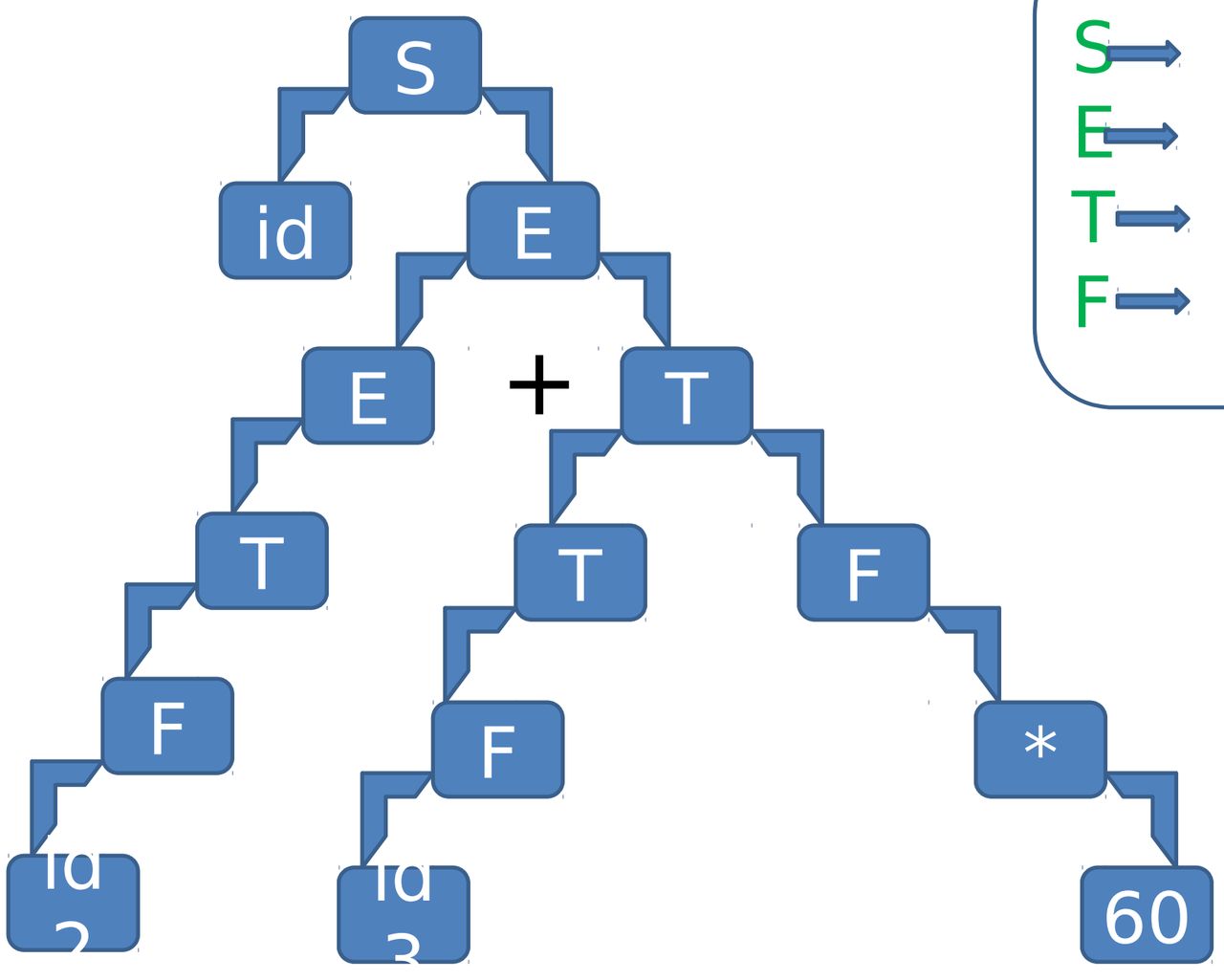
position= initial+rate*60

Lexical analyzer

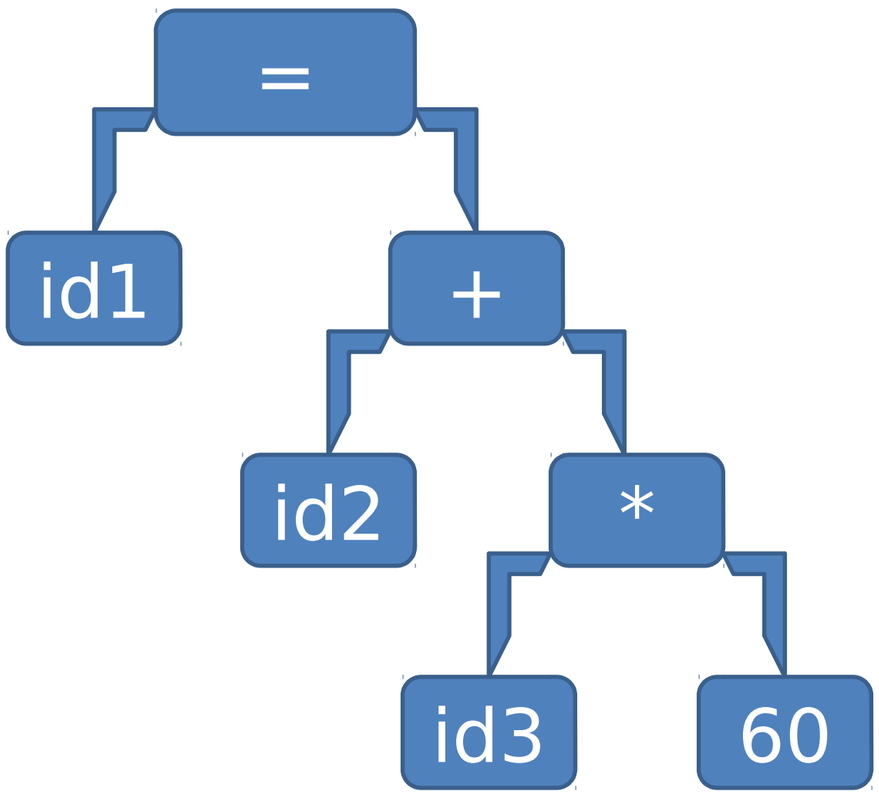
Id1=id2+id3*60

Syntex analyzer (Parser)

Syntex tree

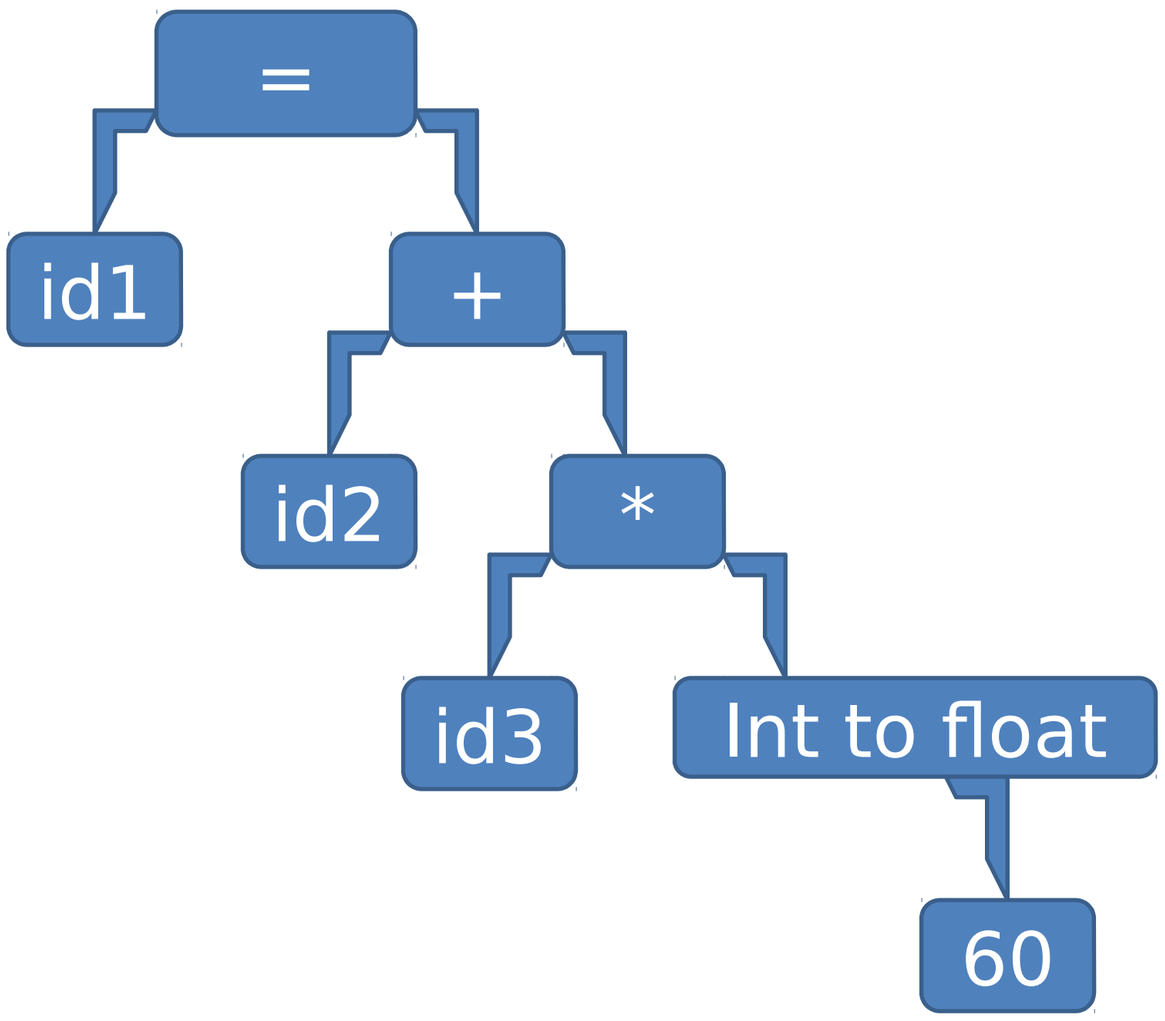


S⟶ id= e
E⟶ E+T\T\
T⟶ T*F/F
F⟶ =id/num

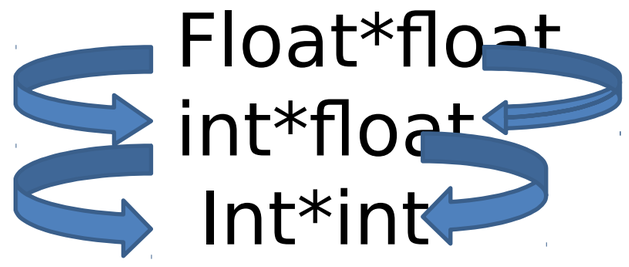


Semantic analyzer

Syntex tree semantically



Int*float
Float id3 *60



Float,position,initial
,rate
Position =
initial+rate*60

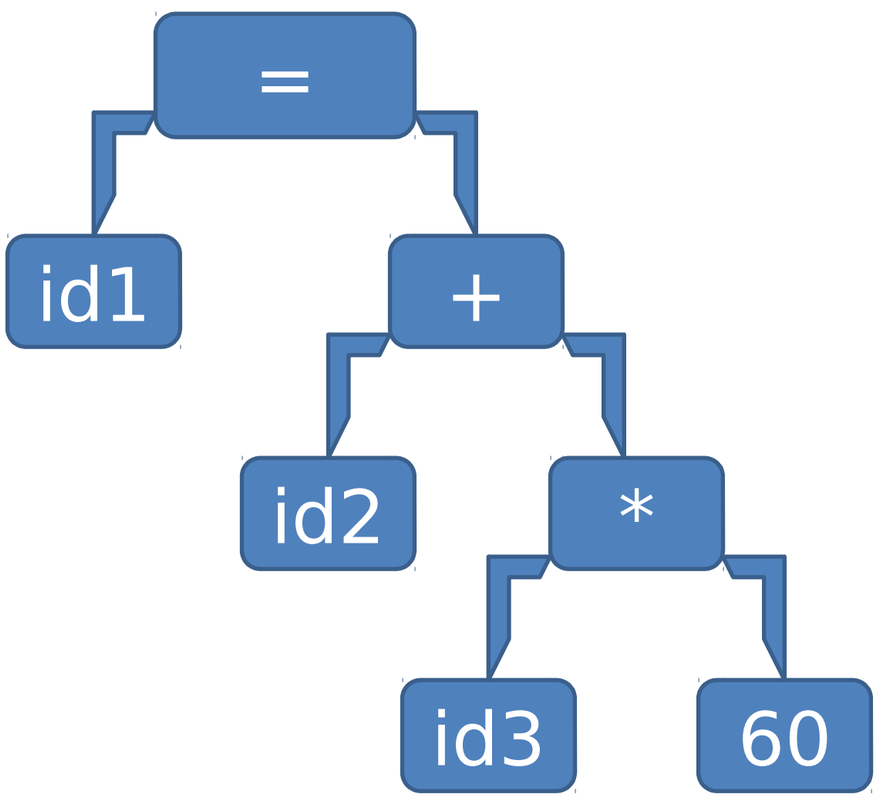
Lexical analyzer

Id1 = id2 + id3 *60

Stream of tokens

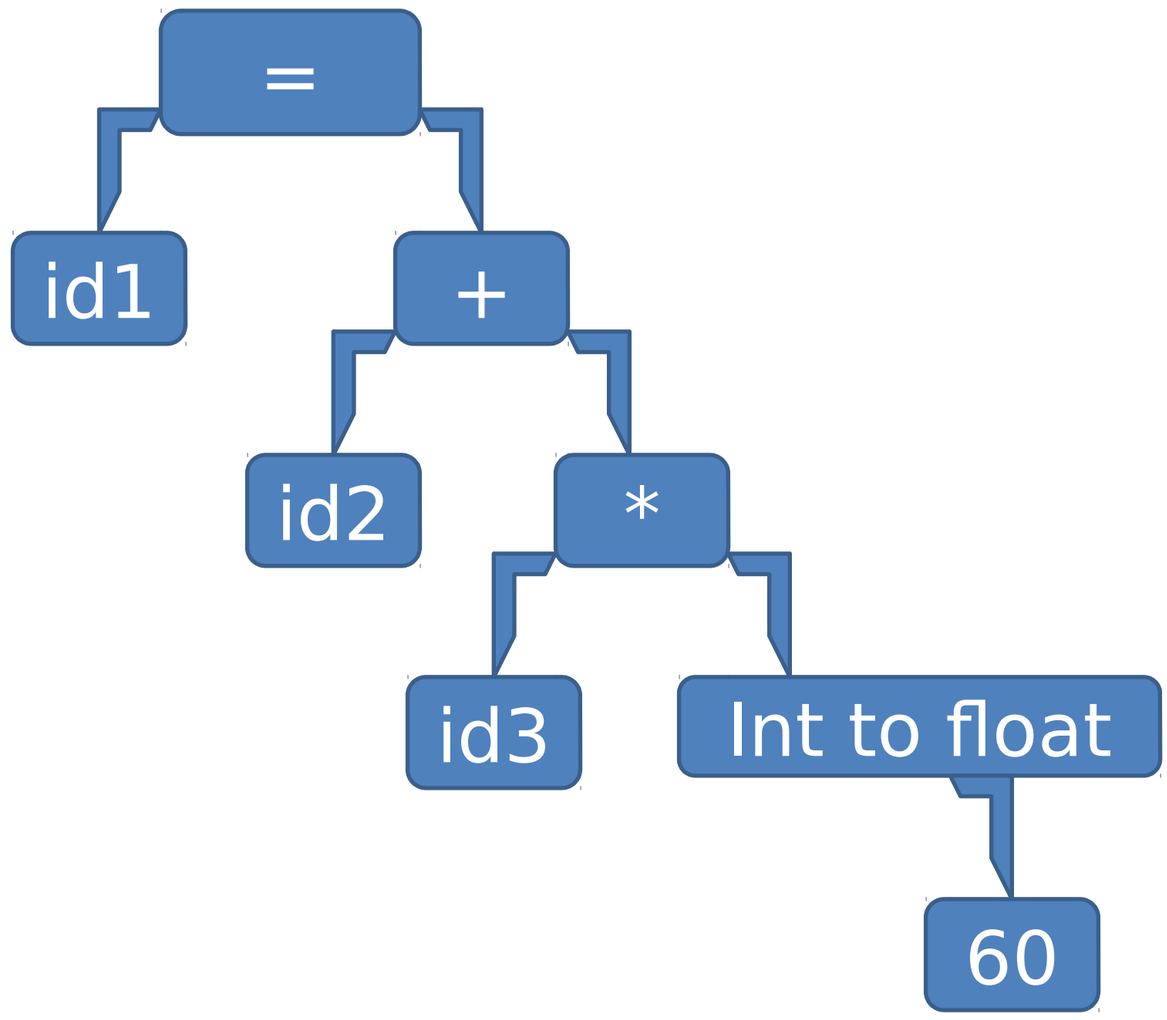
Syntex analyzer

Syntex tree



Semanthet
ic
analyzer

Syntex tree



Intermediate cod
generator

t1 = int to
float 60
T2 = id3*t1
T3 = id2*t2
id`1 = t3

Code
optimizer

t1 = id3*60.0
Id1 = id2 + t1