Front-end constitutes lexical analysis, semantic analysis, syntax analysis, and intermediate code generation.

Lexical analysis is the first phase when compiler scans the source code. This process can be left to right, character by character, and group these characters into tokens.

Using this string example: x+2*y

x => identifier

+ => addition operator

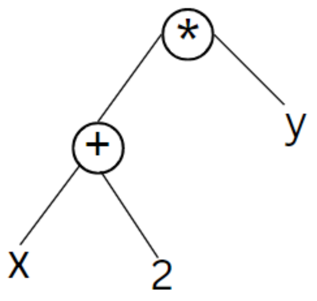
2 => operand

* => multiplication operator

Y => identifier

Syntax analysis obtains tokens from the lexical analyzer, checks if the expression is syntactically correct or not, reports all syntax errors and constructs a hierarchical structure which is known as a parse tree.

Using the example above, our parse tree can be:



 uses the syntax tree of the previous phase along with the symbol table to verify that the given source code is semantically consistent.

For example, using the string above; if its code were as follows.

float x = 20.2;

float y = x*30;

In the above code, the semantic analyzer will typecast the integer 30 to float 30.0 before multiplication.

Once the semantic analysis phase is over the compiler, **generates intermediate** code for the target machine.

For example, x+2*y would translate into:

t1 = y

t2 = 2

t3 = t1 * t2

t4= x

t5 = t3 + t4

solution = t5