Thursday, April 11, 2024 7:53 PM

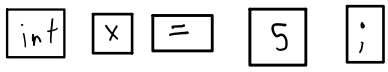
AST Algorithm

Input: array of tokens
Output: "Program" object which is an AST

A section of code can be considered one of several objects defined in the grammar. For example an expression, operator, or a statement.

To build an abstract syntax tree, all that is required is to parse all statements in the file. However, as the first statement is parsed, it will require many other objects to be recursively parsed.

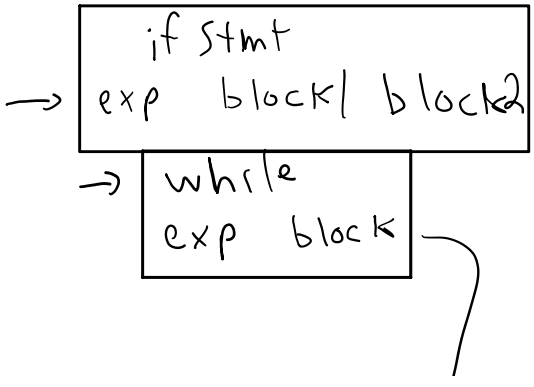
Strategy:
Have a parse method for each grammar object (stmt, exp, op, lhs, etc...)
Each object can have different types (while stmt, if stmt, etc..) so identify which the programmer is attempting by recognizing the correct token
Parse the proper parts of this object
Every parse method should return the token index at which it finished parsing the code object



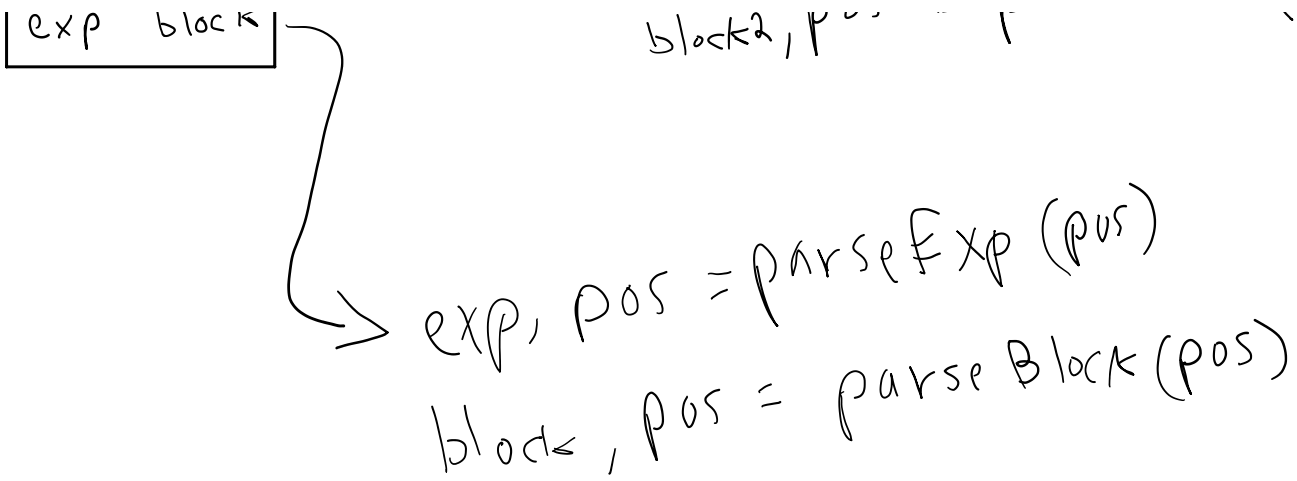

```
parse(tokens)
    position = 0
    while tokens remain
     position=parseStmt(position)
```

```
parseBlock(pos)
    assert token[pos] = {
    while token[pos] != }
     pos=parseStmt(pos)
```

parseStmt → determine stmt type → if, while, etc...




```
exp, pos=parseExp(pos)
block1, pos = parseBlock(pos)
    if token[pos] = Else
      block2, pos = parseBlock(pos)
```


exp block
block2, p
exp, pos = parseExp (pos)
block, pos = parseBlock (pos)