Steven Meyers

Professor Buffenbarger

CS354

12/9/2020

Textbook Assignment 3

1. No, these are not contradictory statements. The programming language having them be left associative does not directly correlate with how they end up in the compiler as the parser could turn the operation into any various forms that the compiler could understand fine. This means that the left-association does not always even make it to the compiler, making it not need to keep association if it produces correct results.
2. Giving unary and binary operators the same level of precedence is quite likely to lead to nonintuitive results. As discussed in this chapter, many programming languages have varying numbers and levels of precedence. Unless the programmer knew how Fortran and Pascal associated the operators as well as how it would operate with other operators, then the code could produce confusing results to people not well versed in the language. This isn't to say that it wouldn't work fine once the programmer knows how the language reads and sets precedence, but until then it could be confusing.
3. The programmer would wish to avoid short circuit evaluation when the conditions cause side effects. For example:

```
A = 1

B  = 2

C = 3

if(A > B && side(20)) {

…

}

boolean side(int i) {

C = 0

Return true;

}

If(C < B) {

Print("Side effect worked!")

}
```

If this program had short circuit evaluation, it would never swap the C value to make the second if statement work as it would just simply eval A > B = false and move on.