Operators and operator precedence in programming language design

Najwani Razali, James Noble and Stuart Marshall
Victoria University of Wellington, New Zealand

October 10, 2014

Operators and operator precedence are common syntactic features in programming languages. Students and experts sometimes make mistakes when using those features in programming language [3, 5, 4, 6]. Such mistakes can be caused by misunderstanding and misinterpretating about the syntax's function. For example, the expression 13 + 5 * 2 has the value of 23 if the expression is evaluated using basic order of evaluation in mathematics, however, has the valueof 36 if the expression is evaluated from left to right by ignoring operator precendence rule.

It is very important to resolve these issues because incorrect understanding of operator precedence concept can also create a security hole in programming. For example, in 2012, the Knight Group Company lost 440 million in the first 30 minutes of their business day. The prediction of the loss may cause by a mistake on operator precedence in their latency cost equation [1].

There are three main goals of this study. Firstly, this research aim to proposed a comprehensive framework of operators and precedence for language designers to use as guidelines in the future. In programming as well as mathematics, evaluation of expression are based on precedence rule. Operators also follow the rule of associativity which is either left-to-right or right-to-left.

Secondly, this reaserch aim to present the easiest way to help students, novices and experts to understand operators and precendence. The proposed framework will indirectly produce a better solution to teach and guide student on operators and precedence. Lastly, the research hopes to present how best to use the existing languages.

Key challenges in undertaking this research relate to designing a solid experimental study for clearly identifying students interpretation of this issue. Some studies have claimed that the use of questionnaires and interview may not be an effective way to gather conceptual understanding [6] while other studies believe they could be useful [2]. Therefore, it is hoped that the discussion following this presentation provides useful insights into the solution of these challenges.

References

[1] *Operator precedence lab*, accessed October 10, 2014. `http://web.stanford.edu/~coopers/securecoding/OperatorPrecedenceLab.pdf`.

[2] Clarke, S. Evaluating a new programming language. In *13th Workshop of the Psychology of Programming Interest Group* (2001), pp. 275–289.

[3] Dolado, J. J., Harman, M., Otero, M. C., and Hu, L. An empirical investigation of the influence of a type of side effects on program comprehension. *Software Engineering, IEEE Transactions on 29*, 7 (2003), 665–670.

[4] Kaczmarczyk, L. C., Petrick, E. R., East, J. P., and Herman, G. L. Identifying student misconceptions of programming. In *Proceedings of the 41st ACM technical symposium on Computer science education* (2010), ACM, pp. 107–111.

[5] Pane, J. F., and Myers, B. A. Tabular and textual methods for selecting objects from a group. In *Visual Languages, 2000. Proceedings. 2000 IEEE International Symposium on* (2000), IEEE, pp. 157–164.

[6] Pillay, N., and Jugoo, V. R. An analysis of the errors made by novice programmers in a first course in procedural programming in java. *Preface of the Editors* (2006), 84.