A Brief History of Choosing First Programming Languages

Leila Goosen

University of Pretoria, South Africa; lgoosen@gk.up.ac.za

Abstract: Choosing the *best* computer language for introducing students to programming is often an emotional issue, leading to protracted debates for many years. This paper aims to document how the development of programming languages has influenced the educational processes of choosing an introductory language since the early days of computing, by exploring some of the "programming languages that have been selected over the last couple of decades and the rational for those selections". [1]

Keywords: History, Selection criteria, First programming language

1 Introduction

Choosing the *best* computer language for introducing students to programming is often an emotional issue, leading to protracted debates over the years [2-6], e.g.

1976: "The selection of languages for use as pedagogical aids in the teaching of computer science is still a big issue at most universities." [7]

1979: "What is a good programming language for beginners?" [8]

1982: "With the diversity of high-level programming languages available, selecting the "right" one for a computer science curriculum or course can be a befuddling process." [6]

1986: "Which computer languages should we be teaching our students - and why?" [9]

1989: "There is increasing discussion about the primary programming language used for undergraduate courses in Computer Science." [10]

So, why is it necessary to pay so much attention to which programming language you start with? [11] The choice of an introductory programming language "is probably the most significant of" the "factors that will shape the competency of the next generation of computer users" [8]. Luker regards "the language used for CS1 and CS2 … as a crucial factor in students' subsequent progress in the discipline" [10].

Please use the following format when citing this chapter:

Goosen, L., 2008, in IFIP International Federation for Information Processing, Volume 269; *History of Computing and Education 3*; John Impagliazzo; (Boston: Springer), pp. 167–170.

2 First Choices

As high-level programming languages became established, faculties, for the first time, needed to start making decisions on which language(s) to implement for teaching [6]: According to [1], they needed to decide whether to move to one of the higher level programming languages, or to continue teaching assembly language programming in introductory courses. Around the mid 1970s, [12] was convinced that the most currently accepted solution" in "to use an existing high-level language".

Another part of the "quarrel" focused around whether a "pedagogical" or a "real" language should be used [12], i.e. languages specifically written for teaching programming, as opposed to languages used in industry. "Is it necessary to teach the languages which are most widely used outside the classroom in order to keep the curriculum relevant to the real world?" [7] According to [13], choosing languages "based on their current *popularity* or the likelihood of their future popularity … has a number of practical benefits." If students, for example, "are constantly reading advertisements for COBOL programmers" [7], they could be more "motivated to study a language that they have heard of" [13]. Similarly, students' interest stems from the fact that they know that a certain language is in demand from "employers, who request that people master the language they will use" [12] in their workplace.

Last, but not the least, [13] also mentions that "a good selection of books and language implementations will be available for a popular language." This brings up text availability as one of the pedagogical factors in support of the process of teaching programming [7]: Especially in introductory courses, students as novice programmers can greatly benefit from the security provided by a readable text. However, if only a small number of texts are available to support a specific language, this may result in the inability to find a suitable text to support a course in that language.

In an effort to facilitate the decision making process [7], in 1976, produced one of the first examples where they applied a set of criteria "to a list of potential languages". In that paper, they also discuss issues with regard to resource constraints, and the influence that cost efficiency could have.

As the "(o)ldest general language in use and the first to become widely used" [14], FORTRAN was most often selected to open "the doors of computing to large numbers of scientists", "mathematicians and engineers who made up computer science faculties at the time" [1]. Compared to other languages of that era, COBOL has a "unique English-like style" [14]. As it was widely used in business data processing, especially departments offering computing courses as part of Business Information Systems programs selected this option. BASIC is "(t)he simplest and one of the most widely used languages" [14]. Because it usually was the only programming language available on personal computers [8], it often became the first language of students at various educational levels.

Giangrande points out that at that stage "there was no named methodology associated with assembly languages, FORTRAN, or COBOL" [1]. Smith criticizes the use of specifically FORTRAN in an introductory course for the effect it had on future programming [7]. As weaknesses in programming languages yet again became a problem, a new methodology, called *structured programming*, emerged. The support of control structures, like while loops, if-then-else statements, etc. for good programming techniques, is something that was lacking in the older languages, e.g. FORTRAN and COBOL [7]. This ensured that these were superseded by a variety of newer programming languages that "incorporated constructs that supported structured programming." [1]

3 The Rise and Fall of Pascal

Until recently (mid 1990s), Pascal used to be the most widely adopted programming language [11] "for introductory computer science courses" [1]. According to [12], one of the principal advantages of Pascal is that it is a simple, small and concise language" specifically designed for teaching structured programming. These "qualities make it usable in a great variety of problems (and not only in numerical calculations), with fair efficiency, and without frustrating restrictions." In 1975, the creator of Pascal himself [15] described the some of the merits of the language "with respect to ease of programming, … efficient implementability" by means of the compiler and interpreter used for the so-called p-code, and easy, practical portability "to a large number of computer systems" [1]. The fact that Pascal and most of the development environments "came with a lot of support and documentation" [11] also made it popular among students.

"During the 1980s, several important languages were created and several languages of the 1970s became popular." [13]. Early in the 1980s [6] mentions some of the comparison criteria for languages such as FORTRAN, COBOL and Pascal. Languages should support "good software engineering practices", as well as showing the existence of control structures to support a preferred programming methodology, adequate diagnostic aids and other programming tools and literature and program libraries for the language.

"(T)he availability of adequate local and vendor support for the implementation of the language" [6] and direct costs with regard to items such as "license fees, and software maintenance contracts" should also be included in calculations.

Although Pascal, BASIC, FORTRAN and COBOL were all abstractions of assembly language [4], that provided big improvements over assembly language, their primary abstraction still required one to think in terms of the structure of the computer, rather than the structure of the problem one was trying to solve. The effort required to perform this mapping, and the fact that it was extrinsic to the programming language, produced programs that were difficult to write and expensive to maintain.

According to [13] "(t)he most striking trend in the field of programming languages" in the 1980s had "been the rise of paradigms, of which the object-oriented paradigm is the best-known." As "support for the creation of objects as instances of a class," [1] function overloading, inheritance and polymorphism became more common, Pascal's popularity gradually began declining - an increasing number of institutions were choosing to introduce undergraduates to programming by teaching object oriented languages, such as C/C++ and Java.

References

1. Giangrande, E.: CS1 programming language options. Journal of Computing Sciences in Colleges. Vol. 22, pp. 153-160 (2007).
2. Ali, A.I., Kohun, F.: Suggested Topics for an IS Introductory Course in Java. In: Proceedings of the Informing Science and Information Technology Education Joint Conference (2005), pp. 33-49, Available via http://pro-ceedings.informingscience.org/InSITE2005/I19f28Ali.pdf. Accessed 25 Apr 2007.
3. Duke, R., Salzman, E., Burmeister, J., Poon, J., Murray, L.: Teaching programming to beginners - choosing the language is just the first step. In: Proceedings of the Australasian Conference on Computing Education. pp.79-86. ACM Press, New York (2000).
4. Goosen, L.: Criteria and guidelines for the selection and implementation of a first programming language in high schools PhD thesis, North-West University (Potchefstroom Campus) (2004); http:// www.puk.ac.za/biblioteek/proefskrifte/2004/goosen_l.pdf
5. Mannila, L., de Raadt, M.: An objective comparison of languages for teaching introductory programming. In: Proceedings of the 6th Baltic Sea conference on Computing education research. pp.32-37. ACM Press, New York (2006).
6. Tharp, A.L.: Selecting the "right" programming language. In: Proceedings of the thirteenth SIGCSE Technical Symposium on Computer Science Education SIGCSE '82. pp. 151-155. ACM Press, New York (1982).
7. Smith, C., Rickman, J.: Selecting languages for pedagogical tools in the computer science curriculum. In: Proceedings of the sixth SIGCSE Technical Symposium on Computer Science Education SIGCSE '76. pp.38-47. ACM Press, New York (1976).
8. Wexelblat, R.L.: First programming language: Consequences (Panel Discussion). In: Proceedings of the 1979 annual conference ACM '79. p.259. ACM Press, New York (1979).
9. Baron, N.S.: The future of computer languages: implications for education. In: Proceedings of the seventeenth SIGCSE Technical Symposium on Computer Science Education SIGCSE '86. pp. 44-49. ACM Press, New York (1986).
10. Luker, P.A.: Never mind the language, what about the paradigm? In: Proceedings of the twentieth SIGCSE Technical Symposium on Computer Science Education SIGCSE '89. pp.252-256. ACM Press, New York (1989).
11. Gupta, D.: What is a good first programming language? Crossroads. Vol. 10, p. 7 (2004).
12. Lecarme, O.: Structured programming, programming teaching and the language Pascal. ACM SIGCSE Bulletin. Vol. 6, pp. 9-15 (1974).
13. King, K.N.: The evolution of the programming languages course. In: Proceedings of the twenty-third SIGCSE technical symposium on Computer science education SIGCSE '92. pp. 213-219. ACM Press, New York (1992).
14. Bergin, T.J.: A history of the history of programming languages. Communications of the ACM. Vol. 50, pp. 69-74 (2007).
15. Wirth, N.: An assessment of the programming language PASCAL. In: Proceedings of the international conference on Reliable software. pp. 23-30. ACM Press, New York (1975).