Informatics in Education, 2024, Vol. 23, No. 4, 719–721
© 2024 *Vilnius University*

Preface

Designing or Choosing Languages for Teaching Programming

Juraj HROMKOVIČ, Dennis KOMM

Department of Computer Science, ETH Zurich, Switzerland
e-mail: juraj.hromkovic@inf.ethz.ch, dennis.komm@inf.ethz.ch

Dedicated to Niklaus Emil Wirth (15 February 1934 – 1 January 2024)

NIKLAUS WIRTH, one of the most influential pioneers of computer science, passed away in Zurich on January 1st of this year. His contributions to software engineering and especially to programming languages are fundamental, unique, and simply amazing. And they have had a huge impact on computer science education worldwide. His concept of simplicity and transparency has changed the style of programming and moved technical coding to enjoyable, creative work. This opened a new dimension in education. Learning by creative work, developing one's own products, and then starting to investigate their functionality and properties in order to work on their improvements in a never-ending loop.

With this special issue we are thanking Niklaus Wirth for his pioneering work in the development of programming languages and in informatics education, and saluting his life's work.

The main goal of this special issue "How to Design or Choose Languages for Teaching Programming" is not to survey the contributions of Wirth to science and engineering, but to honor his unmeasurable impact on programming education and to learn from this for the development of programming languages for schools.

Wirth was a personality who influenced the style of programming as nobody before and after. This was acknowledged by him receiving the Turing award in 1984 "for developing a sequence of innovative computer languages, EULER, ALGOL-W, Pascal, MODULA and Oberon." While the second editor fondly remembers writing Pascal code (after an admittedly rather long episode of producing non-structured and harmfully GOTO-heavy BASIC programs) and sharing it with his friends, the first editor had the privilege of knowing Wirth personally and discussing computer science education with him in particular.

In these discussions, Wirth always underlined the fact that one of his main goals for developing programming languages was to provide novices with well understandable and structured access to learn to program. Interestingly, at the beginning he rejected bringing programming to high schools with the argument that the teachers cannot master this task satisfactorily and that it would be better to start learning programming by professionals at universities to avoid that students develop misconceptions.

Ironically, with developing his probably most famous programming language, named after Blaise Pascal, Wirth himself invented the main ingredient to disprove his "hypothesis," and teaching programming to high school students became a success story in many countries – as attested by some of the contributions to this special issue.

Of course, being a serious scientist who can be convinced by (in this case particularly strong) evidence, Wirth changed his mind and advocated and supported the introduction of computer science to high school curricula. Finally, he even agreed to annually award the *Niklaus Wirth Award* to the best high school projects in Switzerland related to informatics and took part in the celebrations.

Nevertheless, he was quite surprised when we started teaching programming to young kids in primary schools, and he told us that he will observe it from a distance. We were looking forward to discussing this issue with him after successful implementation in Swiss schools during the celebration of his 90th birthday. Unfortunately we did not get the chance.

Wirth became famous for his call for simplicity and clarity in programming (and design in general), and this call is our main message for this special issue on programming education. The contributions to this issue are devoted mainly to the following aspects of teaching to program:

1. To present the history of introducing Pascal to programming courses in high schools and using Pascal in programming competitions, and to explain why Pascal has been so successful and appreciated.
2. To highlight and summarize the merit of the design of Pascal and related languages, and to explain their values in comparison with previous languages.
3. To discuss how to design new programming languages exclusively for education.

Both editors are very thankful for the great privilege to present eight papers that were very carefully selected and each fall into one or more of the above three categories.

Walter Gander was a colleague of Niklaus Wirth at ETH Zurich, as well es a very close friend. In his paper (pages 783–790), he designs a novel recursive algorithm for quadrature in Pascal. In the introduction, he also provides some historical background on the early years of the language. Complementing, Tobias Kohn and Jacqueline Staub, both ETH alumni, take a deep dive into the history of Pascal and its place in the programming language landscape (pages 837–868). In particular, they compare Pascal with Python and analyze their common roots, similarities, and differences.

Two papers discuss the impact that Pascal had in particular in Eastern European countries in the 1980s and 1990s. Valentina Dagienė, Gintautas Grigas, and Tatjana Jevsikova describe (pages 735–765) in a lot of fascinating detail how Pascal shaped early programming education in Lithuania. Maciej Sysło tells the exciting story of the central role Pascal played in Poland (pages 869–882), and why it was the language of choice for many who took part in competitive programming.

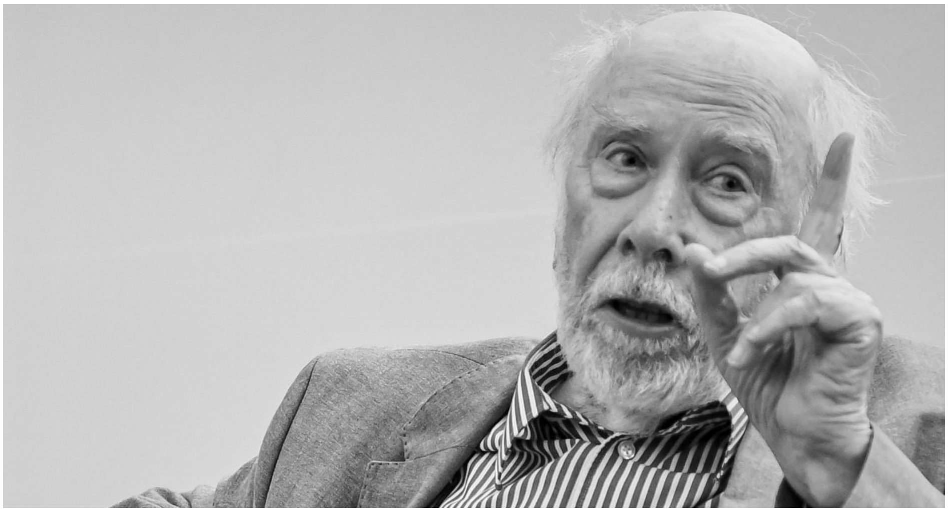
Photo by Andreas Bucher / ETH Zurich

Three of the papers do not directly address Pascal, but introduce programming languages that follow the above call for simplicity. Paul Biberstein, Thomas Castleman, Luming Chen, and Shriram Krishnamurthi present *CODAP Transformers* (pages 723–734), which adds functions to the CODAP programming environment with a focus on data science education. Judith Gal-Ezer and Smadar Szekely introduce the gaming platform *Spark* by *MyQ* (pages 767–781) that aims at fostering computational problem-solving abilities in school students of grades four and beyond. In an autoethnographic paper, Felienne Hermans describes the history of her programming language *Hedy* (pages 791–822), and what drove her design decisions, in particular why her language is built around cognitive load theory.

Finally, Michael Kölling zooms out and looks at the bigger picture of programming education and the design of programming languages (pages 823–836). His conclusion is that there will never be "the" programming language, and that every generation of learners needs their own language. Citing from Kohn's and Staub's paper: "Even Wirth himself hoped that Pascal would make way some day to the next steps in evolution and not become a hindrance to progress itself." But even if everything is just a snapshot and the world keeps spinning, Pascal left a giant mark (and is still used by many), and so did Niklaus Wirth.

Within this special issue, the above papers are ordered by the name of their first authors. We enthusiastically recommend reading all of them, and we promise you will learn new things with every single one. This was the case for us, and it will be the case for you.

Zurich, November 2024 Juraj Hromkovič and Dennis Komm