STM JOURNALS

Recent Trends in
Programming Languages

ISSN: 2455-1821
Volume 12, Issue 1, 2025
January–April
DOI (Journal): 10.37591/RTPL

https://journals.stmjournals.com/rtpl

Review RTPL

The Evolution and Impact of BASIC: The Earliest Computing Language

V. Basil Hans*

Abstract

This study explores the impact of the BASIC programming language, which was designed in the 1960s to make coding accessible to non-specialists. BASIC (Beginner's All-purpose Symbolic Instruction Code) revolutionized computing by democratizing programming, making it possible for hobbyists, students, and budding developers to write their own software. The study highlights how BASIC paved the way for personal computing and laid the foundation for the modern tech landscape. It examines the language's influence on education, its role in shaping the careers of many prominent technologists, and its enduring legacy in computing history. BASIC's simplicity and approachability helped bridge the gap between complex machine languages and user-friendly programming environments, making it one of the most significant tools in the evolution of computer science.

Keywords: BASIC programming language, computing history, design principles, symbols

INTRODUCTION

The BASIC programming language (Beginner's All-purpose Symbolic Instruction Code) is widely regarded as one of the most consequential developments in the history of computing. Developed in 1964 by John G. Kemeny and Thomas E. Kurtz at Dartmouth College, BASIC was created with a singular goal: to make programming accessible to students and non-experts. At a time when coding was reserved for specialists using complex, machine-level languages, BASIC revolutionized the field by offering a simple, easy-to-learn syntax that allowed anyone with a basic understanding of mathematics to start programming [1].

This introduction of BASIC marked the beginning of a profound shift in how people interacted with computers. By lowering the barriers to entry, BASIC played a pivotal role in the rise of personal computing, empowering a generation of hobbyists and enthusiasts to experiment with software. Over time, it became a foundational tool in education, shaping the early careers of many future tech leaders and developers [2].

In this study, we explore how BASIC's design principles, educational impact, and widespread adoption helped shape the modern technology landscape, making it one of the most important programming languages in the history of computing [3].

***Author for Correspondence**
V. Basil Hans
E-mail: vhans2011@gmail.com

Research Professor, Department of Management & Commerce, Srinivas University, Mangaluru, Karnataka, India

Received Date: November 09, 2024
Accepted Date: January 24, 2025
Published Date: February 21, 2025

Citation: V. Basil Hans. The Evolution and Impact of BASIC: The Earliest Computing Language. Recent Trends in Programming Languages. 2025; 12(1): 25–29p.

Programming languages have played a pivotal role in the development of modern information societies. One of the oldest high-level languages is BASIC, a language originally developed to open computers to wider audiences of hobbyists and students. In the early 1960s, computers were anything but "personal". They were specialized devices utilized primarily by mathematicians, scientists, and engineers in industry, government, or academia. Overwhelmingly, users could only access

the machines via punch cards and other intermediary services of the computing staff. Central to the vision of democratizing computing was providing users access to the machine in an interactive mode, enabling them to learn how to use a computer. BASIC sought to do that, enabling non-programmers the ability to write programs to be run on the operating systems of the computing era [4].

BASIC was a central language in the development of the home computer movement of the 1980s. It was estimated in the 1980s that over 95% of all home computers and PC compatibles sold today come with one or multiple versions of BASIC. The combined sales of Commodore, Apple, IBM, and Radio Shack of over 8 million home computers all ran a form of BASIC more or less similar to BERT. Even the Microsoft Windows operating system contained a form of BASIC, called QBasic. There are those who were first introduced to programming through the use of BASIC at a young age and believe BASIC was the perfect beginner's language. In 2015, the last version of BERT for 64-bit operating systems was released and includes GUI rendering, a major change to the original terminal screen. Today, while not as prominent as it once was, BASIC can still be run on Windows, Unix-derived systems, and Unix-proper systems like Linux and FreeBSD. As such, it is fair to remark that BASIC is still in use and continues to have a strong following even today [5].

ORIGINS OF BASIC

The earliest version of BASIC was developed at Dartmouth College in the early 1960s for the GE-225 computer, and nearly from its inception it was made available to other institutions. The users of these early versions were typically college students using teletype terminals and punch card input, who were also likely novices in the practice of computing, just as the designers had intended. The intention to develop BASIC was to facilitate access to computing time across disciplines, for the benefit of everyone from average Joes to experienced scientists. The hope was simply to allow more people to program, so that computing could contribute to society as a whole. These early BASIC implementations were developed for batch processing, making each user completely unaware of every other's computations even if they happened to be simultaneously using the system [6].

Work on Dartmouth BASIC began in 1963 and continued from there. The first version of Dartmouth BASIC became operational on May 1, 1964, on the GE-225 system, and from there moved to other mainframes like the IBM PC. The design of the language was rooted in the dominant philosophy of providing as simple an experience as possible to the user. In particular, it was emphasized allowing the user to determine how much of the computer would be involved in their computation. When the language was introduced at a meeting of educated computing professionals in July 1964, most attendees displayed little interest, not seeing the value in having a user-friendly language. Instead, BASIC was most often picked up by other academic institutions as a way of training non-computer scientists in the art of programming, breaking the stranglehold of the machine-specific languages that had been prevalent up to that point and forever altering the broader landscape of computer science [7].

KEY FEATURES OF BASIC

All computer programming languages face the challenge of being too hard for beginners and too simple for professionals. Writing a computer program should be like constructing a set of instructions for a robot to follow [8]. The writer need not know about the robot's motion controls or vision system. Computer languages can accomplish this objective in two primary ways: by making the language simple to understand and/or by giving the language sophisticated debugging practices. The BASIC language found an excellent balance between these two approaches. It is easy to learn and is a safe first language. Its set of language keywords is few in number and contains many of the essential constructs found in most programming languages, making it a generalist language [9].

Commands can be entered using simple, mnemonic English words with a minimum of punctuation or symbols. The result is that a few simple commands can accomplish quite a bit of work. Second, a person familiar with MTS can do many computer applications using BASIC [10]. Many of the routines

are simple, but some of them are quite sophisticated. Programs are entered as commands, and general symbolic editing is difficult. One of the primary goals in the design of the language was that the program must be easy to enter and simple to modify. For example, it is not necessary for the programmer to specify if variables are integers, reals, or text. BASIC, in fact, freely converts these types. Also, unlike some languages, BASIC can perform arithmetic or solve a logarithm without writing a lot of code. It is suitable for small fragments or single big calculations. It is also a very good fit for teaching because it builds a sound approach. The kinds of problems a novice programmer may solve are limited only by their imagination and grow as more sophisticated tools are learned. Even among people who did not go on to become professional programmers, BASIC indirectly flexed its influence. The BASIC experience created generations of creative, analytic people who went on to become decision makers. These people had a greater understanding of how computers work and, as a result, saw how they could make a machine perform tasks better than they could have done before.

BASIC DIALECTS AND VARIANTS

As mentioned in earlier, over time, several dialects of BASIC were created, growing from seemingly small additions and developments to greatly extended systems. Noteworthy are occasionally even BASIC systems of different businesses that do not have any historical connection. Some of these dialects are described in more detail. Microsoft presented its interpretation of original BASIC in many respects as a *de facto* standard of early home computers. In addition to the numerous home computer and video gaming interpreters of Microsoft BASIC, Microsoft distributed the following ready-for-use BASIC system for DOS and early versions of Windows, having gradually integrated it into DOS:

- *GW-BASIC 6.32:* Introduced by Microsoft for the PC XT in 1983, and supplied with IBM PC ROM-BASIC.
- *QBasic 1.1 6.33:* QBasic was a feature of DOS for PC and x86-based workstations until at least 1995.

The various dialects of the BASIC language have provided enhancements to the initial FOCAL language. New concepts and commands in an interpretation can be regarded as particularly up to date, which points to future developments in the area of computing. Furthermore, the concept of the simple implementation of a self-creating runtime environment in Microsoft interpreters was particularly cutting-edge for a simple programming language, since developers were free to distribute and run source code without time-consuming recompilation. Communally, the additional commands and enhancements in the dialects resulted in increasingly difficult portability between explanations of simple machines. Converting programs between interpreters was often a significant effort. However, at the same time, the increasing number of enthusiasts and hobbyists interested in programming also helped to spread further, bettering the quality of the compilers in question, the wealth of interpreters known as the BASIC variant, and the growing amount of resources.

APPLICATIONS OF BASIC

Beyond the limitations and design trade-offs made by Professor Edwin Martin and the University of Montalo, hoping the dialect will replace Fortran on campus mainframes, BASIC exploded far beyond its original context as a mathematical formula translator. Over its evolution, many new commands were added, and I often found that many respected software developers were somewhat embarrassed by the primitive programming language in which they had once reveled. Indeed, many had expunged part or all of their work on time-shared machines running BASIC years before, as software developers migrated to machine code or other more powerful languages. At the BYTE event in New York, I remember a robust argument among attendees when one onlooker shied slightly at the senior lady from MIT who had written an instruction manual on how to program in BASIC, provoking, from one gentleman, the comical retort something like "BASIC? My dog could create that".

Though BASIC was flawed in the narrowing strategies of its original implementers, their simple goals sprang wide and back. Professor Edwin Martin wrote, "…time-sharing customers want a simple

and practical way of doing problems they now can do by hand, or that may be programmed in a higher-level language. They do not want to spend much time studying how to do what they want to do; they simply want to type the problem and get the solution." Hence, in schools across America, a host of textbooks, programming languages, and software utilities were developed to permit educators to introduce the higher concepts and rudiments of what was, in effect, programming and logic, often as a supplement to mathematics. Calculation and algebra, considered integral to the core curriculum, were taught side by side with programming, manufacturing, and design concepts that even now, some four decades later, have yet to penetrate primary or secondary educational curricula in many countries. Across the USA, individuals young and old were taught the structured commands of workplace computer systems, inculcating a level of human-computer interaction that, through learners, often transformed and immeasurably improved work environments.

IMPACT AND LEGACY OF BASIC

In order to understand the impact and legacy of BASIC, it is essential to remember that BASIC is the first attempt to create a programming language for use that did not need to be compiled or interpreted. As such, it would eventually serve as a template for other types of computing languages. The ancestor of all sorts of interpreted, REPL/CLI-type programming languages like Python, JavaScript, Ruby, PowerShell, and many others is, in fact, BASIC; all the way down to the keystroke behaviors in many cases. In many ways then, the DNA of these kinds of programming languages was first laid down in 1963.

BASIC is also a fundamentally educational programming language. Unlike just about any other programming language, BASIC programming involved actively going out of its way to make it technically less convenient but conceptually easier in order to meet the needs of beginners. These principles also bled directly into the very way computer science is taught and thought of at a popular academic level: a staple of introductory CS classes from the 1960s all the way into the 80s, far exceeding the commercial impact of the language. BASIC is not a movement or company, but it profoundly influenced the education of computer science in the same way that LEGO, Montessori, and Tynker all reflect different approaches to how programming and computer science are taught. Programming has always had the capacity to be turned into an astronomical labor industry, but we still believe that it can be a creative and educational technology. We must understand BASIC in the old-fashioned sense of a technology movement, like early homebrew radio or computer club hobbyists. BASIC was early efforts at popularizing hacking.

CONCLUSION

60 years on, it is not hard to make a case that BASIC and its design significantly shaped the act of learning to program and the languages in which such learning happened. The lowering of the threshold to allow non-programmers to more easily hack the machines came to define the experiences of learning to program, and the languages and practices grew up around the principles established in BASIC, eventually overtaking it.

It seems clear that BASIC’s well-documented instructional aspirations of lowering the bar to entice novice non-programmers to learn how to program bear a significant share of the credit for its legacy. More than simply getting many of a subsequent generation of hackers started on the road to formal programming education, BASIC and its numerous linguistic descendants from the 1960s onward helped to imbue one of the basic commonplaces of contemporary life with a powerful message: that the world of computing and the power to program it could and perhaps should belong more deeply to everyone. Among the implications of its historical context, the reduction of access to programming knowledge did more to reshape the discipline of computing than any other technology of its era. The influence of its design choices can still be found in common code, and the fact that its design choices were made with consideration of non-programmers explains how newcomers have generally been made to experience programming. It is, to be sure, a rather indirect argument for giving BASIC its proper due, but we would hope that the parallel with computing education has at least suggested what is a fertile

ground for future research. Just as efforts to recover the history of programming tools have complicated the narrative of the digital revolution in predictable ways, so too does studying the future look-back transform our understanding of the present that BASIC helped to build.

BASIC's contribution to the world of computing extends far beyond its technical simplicity. As a programming language designed for accessibility, it opened the door for millions of people to engage with computers in a meaningful way, fundamentally altering the trajectory of personal computing. Its user-friendly design empowered students, hobbyists, and early tech enthusiasts to create software without needing deep technical expertise, helping to cultivate a generation of programmers and innovators.

By democratizing programming, BASIC not only facilitated the development of the personal computer revolution but also inspired a lasting culture of experimentation and creativity in software development. Although more modern programming languages have since taken center stage, the legacy of BASIC remains in its enduring influence on how we approach coding education and the democratization of technology. BASIC has left an indelible mark on the tech world, establishing itself as one of the most impactful programming languages in computing history.

REFERENCES

1. Reunanen M, Hoeberechts G, Viljanen T. 'A Step into the Computer Era' – A Comparative Study on Early Home Computing in the United Kingdom, the Netherlands, and Finland. Digital culture, home computers, microcomputers, videogames, history of computing, computer magazines, programming, educational computing. WiderScreen. 2023. Available from: https://widerscreen.fi/assets/ReunanenHoeberechtsViljanen2023.pdf
2. Vinnakota RK, Barr BA, Radavaram S. Open-source 3D printed laboratory for education: illuminating optics and optoelectronics demonstrations. Phys Educ. 2024 Feb 1; 59(2): 025005.
3. Sheffield JL, Parkinson B, Bascom A, Bateman T, Magleby S, Howell LL. Expanding research impact through engaging the maker community and collaborating with digital content creators. PLOS One. 2024 May 8; 19(5): e0302449.
4. Marquardt K, Happe L. Saving bees with computer science: a way to spark enthusiasm and interest through interdisciplinary online courses. In Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. 2023 Jun 29; 145–151.
5. Dakić P. Software compliance in various industries using ci/cd, dynamic microservices, and containers. Open Comput Sci. 2024 Jul 12; 14(1): 20240013.
6. Lovell E, Buechley L, Davis J. LilyTiny in the Wild: Studying the Adoption of a Low-Cost Sewable Microcontroller for Computing Education. In Proceedings of the 2023 ACM Designing Interactive Systems Conference. 2023 Jul 10; 282–293.
7. Vishwakarma S, Baine VS, Mandelbaum R, Kobayashi Y, Lanes O, Wolf-Bauwens ML. The Role of Community Building and Education as Key Pillar of Institutionalizing Responsible Quantum. In 2024 IEEE International Conference on Quantum Computing and Engineering (QCE). 2024 Sep 15; 2: 86–91.
8. Seskir ZC, Migdał P, Weidner C, Anupam A, Case N, Davis N, Decaroli C, Ercan İ, Foti C, Gora P, Jankiewicz K. Quantum games and interactive tools for quantum technologies outreach and education. Opt Eng. 2022 Aug 1; 61(8): 081809.
9. Lovell E, Buechley L, Davis J. The lilytiny: A case study in expanding access to electronic textiles. In CHI Conference on Human Factors in Computing Systems Extended Abstracts. 2022 Apr 27; 1–8.
10. Foley JD, Wallace VL, Chan P. The human factors of computer graphics interaction techniques. IEEE Comput Graph Appl. 1984 Nov; 4(11): 13–48.

