Turiba University

(Chelsia Sengar)

(HISTORY OF PROGRAMMING LANGUAGES)

PROFESSIONAL BACHELOR'S DEGREE

Study programme (Bachelor’s in computer systems)

Author: **(Chelsia Sengar)**

***Thesis* Advisor:** **(Janis Peksa, Dean of IT Dept.)**

2025, RIGA

HISTORY OF PROGRAMMING LANGUAGES

Origins of Programming Languages

The foundation of programming languages began in the 19th century with Ada Lovelace, who is credited with writing the first published computer program for Charles Babbage's Analytical Engine. Earlier mechanical systems, such as the Jacquard Loom and Babbage's Difference Engine, also used simple instruction sets, marking the earliest form of programming languages. Alonzo Church's lambda calculus and Alan Turing's machine model further laid the theoretical groundwork for modern computation.

The First Programming Languages

In the 1940s and early 1950s, the first actual programming languages began to appear. Konrad Zuse's Plankalkül was proposed during WWII, but was never implemented then. John Mauchly's Short Code (1949) was one of the first high-level languages, but required manual conversion to machine code. Autocode, developed by Alick Glennie, became the first compiled language, simplifying programming for machines like the Mark 1 and EDSAC.

FORTRAN and Business-Oriented Languages

In 1954, FORTRAN (Formula Translation) was introduced by John Backus at IBM. It became the first widely used general-purpose high-level programming language, known for its efficiency in scientific computing. Around the same time, Grace Hopper created FLOW-MATIC, the first language to use English-like syntax for business use. It heavily influenced the development of COBOL, which emerged in 1959 and is still used in legacy systems today.

LISP, COBOL, and ALGOL

The late 1950s saw the creation of enduring languages. LISP (1958), by John McCarthy, introduced symbolic computation for AI. COBOL (1959) became dominant in business environments. ALGOL 60, designed by an international committee, introduced key innovations such as block structures, lexical scoping, and the use of Backus–Naur Form (BNF) to formally define syntax, setting the stage for future programming language design.

Algol 68 and the Rise of Simplicity

ALGOL 68 extended ALGOL's concepts with more formal definitions using Van Wijngaarden grammar, higher-order functions, and complex

typing systems. However, its complexity made implementation difficult, leading to criticism. Niklaus Wirth, dissatisfied with ALGOL 68, left the committee and created Pascal, emphasising simplicity and structure—shaping the direction of programming education and future languages.