Programming Languages

Adapted from Wikipedia

B.W.Stuck, MIT: SBEE 1968, SMEE 1969, ScD 1972

Outline

- What Is a Programming Language?
- Procedural Programming
- What Is the Test Problem?
- Object Oriented Programming
- Alfred V Aho
- Historical Survey of Programming Languages

What Is a Programming Language

- A programming language is a system of notation for writing computer programs.
- Programming languages are described in terms of their syntax (form) and semantics (meaning), usually defined by a formal language. Languages usually provide features such as a type system, variables, and mechanisms for error handling.
- An implementation of a programming language is required in order to execute programs, namely an interpreter or a compiler. An interpreter directly executes the source code, while a compiler produces an executable program.
- Thousands of programming languages—often classified as imperative, functional, logic, or object-oriented—have been developed for a wide variety of uses. Many aspects of programming language design involve tradeoffs—for example, exception handling simplifies error handling, but at a performance cost. Programming language theory is the subfield of computer science that studies the design, implementation, analysis, characterization, and classification of programming languages.

Computer Language vs Programming Language

- The term computer language is sometimes used interchangeably with programming language. However, the usage of both terms varies among authors, including the exact scope of each.
- One usage describes programming languages as a subset of computer languages.Similarly, languages used in computing that have a different goal than expressing computer programs are generically designated computer languages.
- For instance, markup languages are sometimes referred to as computer languages to emphasize that they are not meant to be used for programming. One way of classifying computer languages is by the computations they are capable of expressing, as described by the theory of computation.
- The majority of practical programming languages are Turing complete, and all Turing complete languages can implement the same set of algorithms. ANSI/ISO SQL-92 and Charity are examples of languages that are not Turing complete, yet are often called programming languages. However, some authors restrict the term "programming language" to Turing complete languages.

Computer Language vs Programming Language

- Another usage regards programming languages as theoretical constructs for programming abstract machines and computer languages as the subset thereof that runs on physical computers, which have finite hardware resources.
- John C. Reynolds emphasizes that formal specification languages are just as much programming languages as are the languages intended for execution. He also argues that textual and even graphical input formats that affect the behavior of a computer are programming languages, despite the fact they are commonly not Turing-complete, and remarks that ignorance of programming language concepts is the reason for many flaws in input formats.

Procedural Programming

- Procedural programming is a programming paradigm, classified as imperative programming, that involves implementing the behavior of a computer program as procedures (a.k.a. functions, subroutines) that call each other. The resulting program is a series of steps that forms a hierarchy of calls to its constituent procedures.
- The first major procedural programming languages appeared c. 1957–1964, including Fortran, ALGOL, COBOL, PL/I and BASIC. Pascal and C were published c. 1970–1972.
- Computer processors provide hardware support for procedural programming through a stack register and instructions for calling procedures and returning from them. Hardware support for other types of programming is possible, like Lisp machines or Java processors, but no attempt was commercially successful.

What Is the Test Problem?

- When programming languages were first developed, it was felt that the software application program would be developed, and then over time the software would be enhanced as more and more features or capabilities were identified as necessary, but the hardware would not change
- In fact when a critical mass of new features were identified that should be added, that would result in a major new change or new release of the application, it was discovered that the time required to test all the new features to insure no problems with old features led to more and more time spent in testing with each new release
- To quantify this, suppose the initial release of software had 10 states the program could be resident in with one second required to test all combinations of states = 10! = 10x9x8..x1=3,628,800, and the first new release had 2 new states the program could be resident in, then the test had to access 12 states, so the number of possible states tested is 12!=479,001,600 and the test time increased from one second to one hundred thirty two seconds; the next release had 3 new states the program could be resident in, so the test had to access 15!=1.367 * 10^12 states and the test time increased to 360,360 seconds or 100.1 hours!
- Object oriented can address this problem: if the initial release of software had three objects with 4 states, 3 states and 3 states each, then the total number of states is 10, and each module can be tested. If a new release of software adds 2 new states, but no new objects, then the three objects have 5 states, 3 states, and 4 states each, and each of the modules with changes need to be tested, but not the modules with no changes; if a new object with 3 new states is added, then the total number of states is 15! but only the new object need be tested

Object Oriented Programming

- Object-oriented programming (OOP) is a programming paradigm based on the concept of objects, which can contain data and code: data in the form of fields (often known as attributes or properties), and code in the form of procedures (often known as methods). In OOP, computer programs are designed by making them out of objects that interact with one another.
- Many of the most widely used programming languages (such as C++, Java, and Python) are multi-paradigm and support object-oriented programming to a greater or lesser degree, typically in combination with imperative programming, procedural programming and functional programming.
- Significant object-oriented languages include Ada, ActionScript, C++, Common Lisp, C#, Dart, Eiffel, Fortran 2003, Haxe, Java, Kotlin, Logo, MATLAB, Objective-C, Object Pascal, Perl, PHP, Python, R, Raku, Ruby, Scala, SIMSCRIPT, Simula, Smalltalk, Swift, Vala and Visual Basic.NET.

Object Oriented Programming

- Terminology invoking "objects" in the modern sense of object-oriented programming made its first appearance at the artificial intelligence group at MIT in the late 1950s and early 1960s. "Object" referred to LISP atoms with identified properties (attributes). Another early MIT example was Sketchpad created by Ivan Sutherland in 1960–1961; in the glossary of the 1963 technical report based on his dissertation about Sketchpad, Sutherland defined notions of "object" and "instance" (with the class concept covered by "master" or "definition"), albeit specialized to graphical interaction. Also, in 1968, an MIT ALGOL version, AED-0, established a direct link between data structures ("plexes", in that dialect) and procedures, prefiguring what were later termed "messages", "methods", and "member functions". Topics such as data abstraction and modular programming were common points of discussion at this time.
- Independently of later MIT work such as AED, Simula was developed during the years 1961–1967. Simula introduced important concepts that are today an essential part of object-oriented programming, such as class and object, inheritance, and dynamic binding. The object-oriented Simula programming language was used mainly by researchers involved with physical modelling, such as models to study and improve the movement of ships and their content through cargo ports.

Object Oriented Programming

- Influenced by the work at MIT and the Simula language, in November 1966 Alan Kay began working on ideas that would eventually be incorporated into the Smalltalk programming language. Kay used the term "object-oriented programming" in conversation as early as 1967. Although sometimes called "the father of object-oriented programming", Alan Kay has differentiated his notion of OO from the more conventional abstract data type notion of object, and has implied that the computer science establishment did not adopt his notion. A 1976 MIT memo co-authored by Barbara Liskov lists Simula 67, CLU, and Alphard as object-oriented languages, but does not mention Smalltalk.
  - I thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages (so messaging came at the very beginning – it took a while to see how to do messaging in a programming language efficiently enough to be useful). Alan Kay

Object Oriented Programming

- In the 1970s, the first version of the Smalltalk programming language was developed at Xerox PARC by Alan Kay, Dan Ingalls and Adele Goldberg. Smalltalk-72 included a programming environment and was dynamically typed, and at first was interpreted, not compiled. Smalltalk became noted for its application of object orientation at the language-level and its graphical development environment. Smalltalk went through various versions and interest in the language grew. While Smalltalk was influenced by the ideas introduced in Simula 67 it was designed to be a fully dynamic system in which classes could be created and modified dynamically.
- During the late 1970s and 1980s, object-oriented programming rose to prominence. The Flavors object-oriented Lisp was developed starting 1979, introducing multiple inheritance and mixins. In 1981, Goldberg edited the August issue of Byte Magazine, introducing Smalltalk and object-oriented programming to a wide audience. LOOPS, the object system for Interlisp-D, was influenced by Smalltalk and Flavors, and a paper about it was published in 1982. In 1986, the Association for Computing Machinery organized the first Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), which was attended by 1,000 people. Among other developments was the Common Lisp Object System, which integrates functional programming and object-oriented programming and allows extension via a Meta-object protocol. In the 1980s, there were a few attempts to design processor architectures that included hardware support for objects in memory but these were not successful. Examples include the Intel iAPX 432 and the Linn Smart Rekursiv.

Object Oriented Programming

- In the mid-1980s Objective-C was developed by Brad Cox, who had used Smalltalk at ITT Inc.. Bjarne Stroustrup, who had used Simula for his PhD thesis, created the object-oriented C++. In 1985, Bertrand Meyer also produced the first design of the Eiffel language. Focused on software quality, Eiffel is a purely object-oriented programming language and a notation supporting the entire software lifecycle. Meyer described the Eiffel software development method, based on a small number of key ideas from software engineering and computer science, in Object-Oriented Software Construction. Essential to the quality focus of Eiffel is Meyer's reliability mechanism, design by contract, which is an integral part of both the method and language.
- In the early and mid-1990s object-oriented programming developed as the dominant programming paradigm when programming languages supporting the techniques became widely available. These included Visual FoxPro 3.0,C++, and Delphi. Its dominance was further enhanced by the rising popularity of graphical user interfaces, which rely heavily upon object-oriented programming techniques. An example of a closely related dynamic GUI library and OOP language can be found in the Cocoa frameworks on Mac OS X, written in Objective-C, an object-oriented, dynamic messaging extension to C based on Smalltalk. OOP toolkits also enhanced the popularity of event-driven programming (although this concept is not limited to OOP).

Object Oriented Programming

- At ETH Zürich, Niklaus Wirth and his colleagues investigated the concept of type checking across module boundaries. Modula-2 (1978) included this concept, and their succeeding design, Oberon (1987), included a distinctive approach to object orientation, classes, and such. Inheritance is not obvious in Wirth's design since his nomenclature looks in the opposite direction: It is called type extension and the viewpoint is from the parent down to the inheritor.
- Object-oriented features have been added to many previously existing languages, including Ada, BASIC, Fortran, Pascal, and COBOL. Adding these features to languages that were not initially designed for them often led to problems with compatibility and maintainability of code.
- More recently, some languages have emerged that are primarily object-oriented, but that are also compatible with procedural methodology. Two such languages are Python and Ruby. Probably the most commercially important recent object-oriented languages are Java, developed by Sun Microsystems, as well as C# and Visual Basic.NET (VB.NET), both designed for Microsoft's .NET platform. Each of these two frameworks shows, in its way, the benefit of using OOP by creating an abstraction from implementation. VB.NET and C# support cross-language inheritance, allowing classes defined in one language to subclass classes defined in the other language.

Object Oriented Programming

- Object-oriented programming uses objects, but not all of the associated techniques and structures are supported directly in languages that claim to support OOP. The features listed below are common among languages considered to be strongly class- and object-oriented (or multi-paradigm with OOP support), with notable exceptions mentioned.
- Christopher J. Date stated that critical comparison of OOP to other technologies, relational in particular, is difficult because of lack of an agreed-upon and rigorous definition of OOP.
- Shared with non-OOP Languages
  - **Variables** that can store information formatted in a small number of built-in data types like integers and alphanumeric characters. This may include data structures like strings, lists, and hash tables that are either built-in or result from combining variables using memory pointers.
  - **Procedures** – also known as **functions, methods, routines**, or **subroutines** – that take input, generate output, and manipulate data. Modern languages include structured programming constructs like loops and conditionals.
- Modular programming support provides the ability to group procedures into files and modules for organizational purposes. Modules are namespaced so identifiers in one module will not conflict with a procedure or variable sharing the same name in another file or module

Object Oriented Programming

- An object is a data structure or abstract data type containing fields (state variables containing data) and methods (subroutines or procedures defining the object's behavior in code). Fields may also be known as members, attributes, or properties. Objects are typically stored as contiguous regions of memory. Objects are accessed somewhat like variables with complex internal structures, and in many languages are effectively pointers, serving as actual references to a single instance of said object in memory within a heap or stack.
- Objects sometimes correspond to things found in the real world. For example, a graphics program may have objects such as "circle", "square", and "menu". An online shopping system might have objects such as "shopping cart", "customer", and "product". Sometimes objects represent more abstract entities, like an object that represents an open file, or an object that provides the service of translating measurements from U.S. customary to metric.
- Objects can contain other objects in their instance variables; this is known as object composition. For example, an object in the Employee class might contain (either directly or through a pointer) an object in the Address class, in addition to its own instance variables like "first_name" and "position". Object composition is used to represent "has-a" relationships: every employee has an address, so every Employee object has access to a place to store an Address object (either directly embedded within itself or at a separate location addressed via a pointer). Date and Darwen have proposed a theoretical foundation that uses OOP as a kind of customizable type system to support RDBMS, but it forbids object pointers.

Alfred V Aho

- Aho received a B.A.Sc. (1963) in Engineering Physics from the University of Toronto, then an M.A. (1965) and Ph.D. (1967) in Electrical Engineering/Computer Science from Princeton University.[6] He conducted research at Bell Labs from 1967 to 1991, and again from 1997 to 2002 as Vice President of the Computing Sciences Research Center. Since 1995, he has held the Lawrence Gussman Professorship in Computer Science at Columbia University. He served as chair of the department from 1995 to 1997, and again in the spring of 2003.
- In his PhD thesis Aho created indexed grammars[9] and the nested-stack automaton as vehicles for extending the power of context-free languages, but retaining many of their decidability and closure properties. One application of indexed grammars is modelling parallel rewriting systems, particularly in biological applications.
- After graduating from Princeton, Aho joined the Computing Sciences Research Center at Bell Labs where he devised efficient regular expression and string-pattern matching algorithms that he implemented in the first versions of the Unix tools egrep and fgrep. The fgrep algorithm has become known as the Aho–Corasick algorithm; it is used by several bibliographic search-systems, including the one developed by Margaret J. Corasick, and by other string-searching applications.
- At Bell Labs, Aho worked closely with Steve Johnson and Jeffrey Ullman to develop efficient algorithms for analyzing and translating programming languages. Steve Johnson used the bottom-up LALR parsing algorithms to create the syntax-analyzer generator yacc, and Michael E. Lesk and Eric Schmidt used Aho's regular-expression pattern-matching algorithms to create the lexical-analyzer generator lex. The lex and yacc tools and their derivatives have been used to develop the front ends of many of today's programming language compilers.

Alfred V Aho

- Aho and Ullman wrote a series of textbooks on compiling techniques that codified the theory relevant to compiler design. Their 1977 textbook Principles of Compiler Design had a green dragon on the front cover and became known as "the green dragon book". In 1986 Aho and Ullman were joined by Ravi Sethi to create a new edition, "the red dragon book" (which was briefly shown in the 1995 movie Hackers), and in 2006 also by Monica Lam to create "the purple dragon book". The dragon books are used for university courses as well as industry references.
- In 1974, Aho, John Hopcroft, and Ullman wrote The Design and Analysis of Computer Algorithms, codifying some of their early research on algorithms. This book became one of the most highly cited books in computer science for several decades and helped to stimulate the creation of algorithms and data structures as a central course in the computer science curriculum.
- Aho is also widely known for his co-authorship of the AWK programming language with Peter J. Weinberger and Brian Kernighan (the "A" stands for "Aho"). As of 2010 Aho's research interests include programming languages, compilers, algorithms, and quantum computing. He is part of the Language and Compilers research-group at Columbia University.
- Overall, his works have been cited 81,040 times and he has an h-index of 66, as of May 8, 2019.

Fortran

- In late 1953, John W. Backus submitted a proposal to his superiors at IBM to develop a more practical alternative to assembly language for programming their IBM 704 mainframe computer. Backus' historic FORTRAN team consisted of programmers Richard Goldberg, Sheldon F. Best, Harlan Herrick, Peter Sheridan, Roy Nutt, Robert Nelson, Irving Ziller, Harold Stern, Lois Haibt, and David Sayre Its concepts included easier entry of equations into a computer, an idea developed by J. Halcombe Laning and demonstrated in the Laning and Zierler system of 1952.
- The Fortran Automatic Coding System for the IBM 704 (October 15, 1956), the first programmer's reference manual for Fortran
- A draft specification for The IBM Mathematical Formula Translating System was completed by November 1954. The first manual for FORTRAN appeared in October 1956, with the first FORTRAN compiler delivered in April 1957. Fortran produced efficient enough code for assembly language programmers to accept a high-level programming language replacement.
- John Backus said during a 1979 interview with Think, the IBM employee magazine, "Much of my work has come from being lazy. I didn't like writing programs, and so, when I was working on the IBM 701, writing programs for computing missile trajectories, I started work on a programming system to make it easier to write programs."
- The language was widely adopted by scientists for writing numerically intensive programs, which encouraged compiler writers to produce compilers that could generate faster and more efficient code. The inclusion of a complex number data type in the language made Fortran especially suited to technical applications such as electrical engineering.

Fortran

- Fortran (/ˈfɔːrtræn/; formerly FORTRAN) Formula Translating System is a third generation, compiled, imperative programming language that is especially suited to numeric computation and scientific computing.
- Fortran was originally developed by IBM.It first compiled correctly in 1958.Fortran computer programs have been written to support scientific and engineering applications, such as numerical weather prediction, finite element analysis, computational fluid dynamics, geophysics, computational physics, crystallography and computational chemistry. It is a popular language for high-performance computing[and is used for programs that benchmark and rank the world's fastest supercomputers.
- The IBM Blue Gene/P supercomputer installation in 2007 at the Argonne Leadership Angela Yang Computing Facility located in the Argonne National Laboratory, in Lemont, Illinois, US
- Fortran has evolved through numerous versions and dialects. In 1966, the American National Standards Institute (ANSI) developed a standard for Fortran to limit proliferation of compilers using slightly different syntax.[8] Successive versions have added support for a character data type (Fortran 77), structured programming, array programming, modular programming, generic programming (Fortran 90), parallel computing (Fortran 95), object-oriented programming (Fortran 2003), and concurrent programming (Fortran 2008).
- Since April 2024, Fortran has ranked among the top ten languages in the TIOBE index, a measure of the popularity of programming languages.[9]

Programming Language 1 PL1

- PL/I (Programming Language One, pronounced /piː ɛl wʌn/ and sometimes written PL/1) is a procedural, imperative computer programming language initially developed by IBM. It is designed for scientific, engineering, business and system programming. It has been in continuous use by academic, commercial and industrial organizations since it was introduced in the 1960s.
- The PL/1 ANSI standard, X3.53-1976, was published in 1976.
- PL/I's main domains are data processing, numerical computation, scientific computing, and system programming. It supports recursion, structured programming, linked data structure handling, fixed-point, floating-point, complex, character string handling, and bit string handling. The language syntax is English-like and suited for describing complex data formats with a wide set of functions available to verify and manipulate them.

Programming Language 1

- The goals for PL/I evolved during the early development of the language. Competitiveness with COBOL's record handling and report writing was required. The language's scope of usefulness grew to include system programming and event-driven programming. Additional goals for PL/I were:
  - Performance of compiled code competitive with that of Fortran (but this was not achieved)[citation needed]
  - Extensibility for new hardware and new application areas
  - Improved productivity of the programming process, transferring effort from the programmer to the compiler
  - Machine independence to operate effectively on the main computer hardware and operating systems
- To achieve these goals, PL/I borrowed ideas from contemporary languages while adding substantial new capabilities and casting it with a distinctive concise and readable syntax. Many principles and capabilities combined to give the language its character and were important in meeting the language's goals:
  - Block structure, with underlying semantics (including recursion), similar to Algol 60. Arguments are passed using call by reference, using dummy variables for values where needed (call by value).
  - A wide range of computational data types, program control data types, and forms of data structure (strong typing).
  - Dynamic extents for arrays and strings with inheritance of extents by procedure parameters.
  - Concise syntax for expressions, declarations, and statements with permitted abbreviations. Suitable for a character set of 60 glyphs and sub-settable to 48.
  - An extensive structure of defaults in statements, options, and declarations to hide some complexities and facilitate extending the language while minimizing keystrokes.
  - Powerful iterative processing with good support for structured programming.
- There were to be no reserved words (although the function names DATE and TIME initially proved to be impossible[citation needed] to meet this goal). New attributes, statements and statement options could be added to PL/I without invalidating existing programs. Not even IF, THEN, ELSE, and DO were reserved.
  - Orthogonality: each capability to be independent of other capabilities and freely combined with other capabilities wherever meaningful. Each capability to be available in all contexts where meaningful, to exploit it as widely as possible and to avoid "arbitrary restrictions". Orthogonality helps make the language "large".
  - Exception handling capabilities for controlling and intercepting exceptional conditions at run time.
  - Programs divided into separately compilable sections, with extensive compile-time facilities (a.k.a. macros), not part of the standard, for tailoring and combining sections of source code into complete programs. External names to bind separately compiled procedures into a single program.
  - Debugging facilities integrated into the language.

Basic Programming Language

- BASIC (Beginners' All-purpose Symbolic Instruction Code)[1] is a family of general-purpose, high-level programming languages designed for ease of use. The original version was created by John G. Kemeny and Thomas E. Kurtz at Dartmouth College in 1963. They wanted to enable students in non-scientific fields to use computers. At the time, nearly all computers required writing custom software, which only scientists and mathematicians tended to learn.

- In addition to the programming language, Kemeny and Kurtz developed the Dartmouth Time Sharing System (DTSS), which allowed multiple users to edit and run BASIC programs simultaneously on remote terminals. This general model became popular on minicomputer systems like the PDP-11 and Data General Nova in the late 1960s and early 1970s. Hewlett-Packard produced an entire computer line for this method of operation, introducing the HP2000 series in the late 1960s and continuing sales into the 1980s. Many early video games trace their history to one of these versions of BASIC.

- The emergence of microcomputers in the mid-1970s led to the development of multiple BASIC dialects, including Microsoft BASIC in 1975. Due to the tiny main memory available on these machines, often 4 KB, a variety of Tiny BASIC dialects were also created. BASIC was available for almost any system of the era, and became the de facto programming language for home computer systems that emerged in the late 1970s. These PCs almost always had a BASIC interpreter installed by default, often in the machine's firmware or sometimes on a ROM cartridge.

- BASIC declined in popularity in the 1990s, as more powerful microcomputers came to market and programming languages with advanced features (such as Pascal and C) became tenable on such computers. By then, most nontechnical personal computer users relied on pre-written applications rather than writing their own programs. In 1991, Microsoft released Visual Basic, combining an updated version of BASIC with a visual forms builder. This reignited use of the language and "VB" remains a major programming language in the form of VB.NET, while a hobbyist scene for BASIC more broadly continues to exist.[

Basic Programming Language

- John G. Kemeny was the chairman of the Dartmouth College Mathematics Department. Based largely on his reputation as an innovator in math teaching, in 1959 the College won an Alfred P. Sloan Foundation award for $500,000 to build a new department building. Thomas E. Kurtz had joined the department in 1956, and from the 1960s Kemeny and Kurtz agreed on the need for programming literacy among students outside the traditional STEM fields. Kemeny later noted that "Our vision was that every student on campus should have access to a computer, and any faculty member should be able to use a computer in the classroom whenever appropriate. It was as simple as that."

- Kemeny and Kurtz had made two previous experiments with simplified languages, DARSIMCO (Dartmouth Simplified Code) and DOPE (Dartmouth Oversimplified Programming Experiment). These did not progress past a single freshman class. New experiments using Fortran and ALGOL followed, but Kurtz concluded these languages were too tricky for what they desired. As Kurtz noted, Fortran had numerous oddly formed commands, notably an "almost impossible-to-memorize convention for specifying a loop: DO 100, I = 1, 10, 2. Is it '1, 10, 2' or '1, 2, 10', and is the comma after the line number required or not?"

- Moreover, the lack of any sort of immediate feedback was a key problem; the machines of the era used batch processing and took a long time to complete a run of a program. While Kurtz was visiting MIT, John McCarthy suggested that time-sharing offered a solution; a single machine could divide up its processing time among many users, giving them the illusion of having a (slow) computer to themselves.[8] Small programs would return results in a few seconds. This led to increasing interest in a system using time-sharing and a new language specifically for use by non-STEM students.

- Kemeny wrote the first version of BASIC. The acronym BASIC comes from the name of an unpublished paper by Thomas Kurtz. The new language was heavily patterned on FORTRAN II; statements were one-to-a-line, numbers were used to indicate the target of loops and branches, and many of the commands were similar or identical to Fortran. However, the syntax was changed wherever it could be improved. For instance, the difficult to remember DO loop was replaced by the much easier to remember FOR I = 1 TO 10 STEP 2, and the line number used in the DO was instead indicated by the NEXT I.[a] Likewise, the cryptic IF statement of Fortran, whose syntax matched a particular instruction of the machine on which it was originally written, became the simpler IF I=5 THEN GOTO 100. These changes made the language much less idiosyncratic while still having an overall structure and feel similar to the original FORTRAN.

C Programming Language

- C (pronounced /ˈsiː/ – like the letter c) is a general-purpose programming language. It was created in the 1970s by Dennis Ritchie and remains very widely used and influential. By design, C's features cleanly reflect the capabilities of the targeted CPUs. It has found lasting use in operating systems code (especially in kernels[7]), device drivers, and protocol stacks, but its use in application software has been decreasing. C is commonly used on computer architectures that range from the largest supercomputers to the smallest microcontrollers and embedded systems.

- A successor to the programming language B, C was originally developed at Bell Labs by Ritchie between 1972 and 1973 to construct utilities running on Unix. It was applied to re-implementing the kernel of the Unix operating system. During the 1980s, C gradually gained popularity. It has become one of the most widely used programming languages, with C compilers available for practically all modern computer architectures and operating systems. The book The C Programming Language, co-authored by the original language designer, served for many years as the de facto standard for the language. C has been standardized since 1989 by the American National Standards Institute (ANSI) and, subsequently, jointly by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC).

- C is an imperative procedural language, supporting structured programming, lexical variable scope, and recursion, with a static type system. It was designed to be compiled to provide low-level access to memory and language constructs that map efficiently to machine instructions, all with minimal runtime support. Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant C program written with portability in mind can be compiled for a wide variety of computer platforms and operating systems with few changes to its source code.

- Since 2000, C has consistently ranked among the top three languages in the TIOBE index, a measure of the popularity of programming languages.

C Programming Language

- C is an imperative, procedural language in the ALGOL tradition. It has a static type system. In C, all executable code is contained within subroutines (also called "functions", though not in the sense of functional programming). Function parameters are passed by value, although arrays are passed as pointers, i.e. the address of the first item in the array. Pass-by-reference is simulated in C by explicitly passing pointers to the thing being referenced.
- C program source text is free-form code. Semicolons terminate statements, while curly braces are used to group statements into blocks.
- The C language also exhibits the following characteristics:
  - The language has a small, fixed number of keywords, including a full set of control flow primitives: if/else, for, do/while, while, and switch. User-defined names are not distinguished from keywords by any kind of sigil.
  - It has a large number of arithmetic, bitwise, and logic operators: +,+=,++,&,||, etc.
  - More than one assignment may be performed in a single statement.
- Functions:
  - Function return values can be ignored, when not needed.
  - Function and data pointers permit ad hoc run-time polymorphism.
  - Functions may not be defined within the lexical scope of other functions.
  - Variables may be defined within a function, with scope.
  - A function may call itself, so recursion is supported.
- Data typing is static, but weakly enforced; all data has a type, but implicit conversions are possible.
- User-defined (typedef) and compound types are possible.
- Heterogeneous aggregate data types (struct) allow related data elements to be accessed and assigned as a unit. The contents of whole structs cannot be compared using a single built-in operator (the elements must be compared individually).
- Union is a structure with overlapping members; it allows multiple data types to share the same memory location.

C Programming Language

- Array indexing is a secondary notation, defined in terms of pointer arithmetic. Whole arrays cannot be assigned or compared using a single built-in operator. There is no "array" keyword in use or definition; instead, square brackets indicate arrays syntactically, for example month
  - Enumerated types are possible with the enum keyword. They are freely interconvertible with integers.
  - Strings are not a distinct data type, but are conventionally implemented as null-terminated character arrays.
- Low-level access to computer memory is possible by converting machine addresses to pointers.
- Procedures (subroutines not returning values) are a special case of function, with an empty return type void.
- Memory can be allocated to a program with calls to library routines.
- A preprocessor performs macro definition, source code file inclusion, and conditional compilation.
- There is a basic form of modularity: files can be compiled separately and linked together, with control over which functions and data objects are visible to other files via static and extern attributes.
  - Complex functionality such as I/O, string manipulation, and mathematical functions are consistently delegated to library routines.
  - The generated code after compilation has relatively straightforward needs on the underlying platform, which makes it suitable for creating operating systems and for use in embedded systems.
  - While C does not include certain features found in other languages (such as object orientation and garbage collection), these can be implemented or emulated, often through the use of external libraries (e.g., the GLib Object System or the Boehm garbage collector).

C++ Programming Language

- C++ (/ˈsiː plʌs plʌs/, pronounced "C plus plus" and sometimes abbreviated as CPP) is a high-level, general-purpose programming language created by Danish computer scientist Bjarne Stroustrup. First released in 1985 as an extension of the C programming language, it has since expanded significantly over time; as of 1997, C++ has object-oriented, generic, and functional features, in addition to facilities for low-level memory manipulation for systems like microcomputers or to make operating systems like Linux or Windows. It is usually implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Embarcadero, Oracle, and IBM.
- C++ was designed with systems programming and embedded, resource-constrained software and large systems in mind, with performance, efficiency, and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including desktop applications, video games, servers (e.g., e-commerce, web search, or databases), and performance-critical applications (e.g., telephone switches or space probes).

MATLAB Programming Language

- MATLAB (an abbreviation of "MATrix LABoratory") is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.
- Although MATLAB is intended primarily for numeric computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.
- As of 2020, MATLAB has more than four million users worldwide. They come from various backgrounds of engineering, science, and economics. As of 2017, more than 5000 global colleges and universities use MATLAB to support instruction and research.

MATLAB Programming Language

- MATLAB was invented by mathematician and computer programmer Cleve Moler. The idea for MATLAB was based on his 1960s PhD thesis. Moler became a math professor at the University of New Mexico and started developing MATLAB for his students as a hobby. He developed MATLAB's initial linear algebra programming in 1967 with his one-time thesis advisor, George Forsythe. This was followed by Fortran code for linear equations in 1971.
- Before version 1.0, MATLAB "was not a programming language; it was a simple interactive matrix calculator. There were no programs, no toolboxes, no graphics. And no ODEs or FFTs."
- The first early version of MATLAB was completed in the late 1970s. The software was disclosed to the public for the first time in February 1979 at the Naval Postgraduate School in California. Early versions of MATLAB were simple matrix calculators with 71 pre-built functions. At the time, MATLAB was distributed for free to universities. Moler would leave copies at universities he visited and the software developed a strong following in the math departments of university campuses.
- In the 1980s, Cleve Moler met John N. Little. They decided to reprogram MATLAB in C and market it for the IBM desktops that were replacing mainframe computers at the time. John Little and programmer Steve Bangert re-programmed MATLAB in C, created the MATLAB programming language, and developed features for toolboxes.

Java

- Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need to recompile. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but has fewer low-level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages.
- Java gained popularity shortly after its release, and has been a very popular programming language since then. Java was the third most popular programming language in 2022 according to GitHub. Although still widely popular, there has been a gradual decline in use of Java in recent years with other languages using JVM gaining popularity.
- Java was originally developed by James Gosling at Sun Microsystems. It was released in May 1995 as a core component of Sun's Java platform. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun had relicensed most of its Java technologies under the GPL-2.0-only license. Oracle offers its own HotSpot Java Virtual Machine, however the official reference implementation is the OpenJDK JVM which is free open-source software and used by most developers and is the default JVM for almost all Linux distributions.

Python

- Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.
- Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.
- Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0.[35] Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2.
- Python consistently ranks as one of the most popular programming languages, and has gained widespread use in the machine learning community.

Python Programming Language

- Python was invented in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system.
- Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's "benevolent dictator for life" (BDFL), a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker (he's since come out of retirement and is self-titled "BDFL-emeritus"). In January 2019, active Python core developers elected a five-member Steering Council to lead the project.
- Python 2.0 was released on 16 October 2000, with many major new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0 was released on 3 December 2008, with many of its major features backported to Python 2.6.x[48] and 2.7.x. Releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3

RUST

- Rust is a general-purpose programming language emphasizing performance, type safety, and concurrency. It enforces memory safety, meaning that all references point to valid memory, without a garbage collector. To simultaneously enforce memory safety and prevent data races, its "borrow checker" tracks the object lifetime of all references in a program during compiling.
- Rust was influenced by ideas from functional programming, including immutability, higher-order functions, and algebraic data types. It is popular for systems programming. Rust does not enforce a programming paradigm, but supports object-oriented programming via structs, enums, traits, and methods, and supports functional programming via immutability, pure functions, higher order functions, and pattern matching.
- Software developer Graydon Hoare created Rust as a personal project while working at Mozilla Research in 2006. Mozilla officially sponsored the project in 2009. In the years following the first stable release in May 2015, Rust was adopted by companies including Amazon, Discord, Dropbox, Google (Alphabet), Meta, and Microsoft. In December 2022, it became the first language other than C and assembly to be supported in the development of the Linux kernel.
- Rust has been noted for its rapid adoption, and has been studied in programming language theory research.

RUST Programming Language

- Memory safety: Rust is built with a strong focus on memory safety, which helps prevent common programming errors.
- Concurrency: Rust offers great features for concurrent programming, making it a good choice for writing reliable and parallel applications.
- Ecosystem: The Rust ecosystem includes tens of thousands of crates, or Rust code libraries, that are available on crates.io.
- Error handling: Rust's error handling is powerful and uses built-in Option and Result types to indicate when a return value may be missing or an error may have occurred.
- Low memory footprint: Rust has a small runtime footprint, making it a good choice for building applications for microcontrollers, embedded systems, and IoT devices.
- Targets bare-metal: Rust can be used to write device drivers or OS kernels, and can be used as a "high level assembler".