Technical and Vocational University

Minab Branch

Course

Programming Languages Design and Laboratory

Lecturer

Mohammad Ahmadzadeh

Fall of 2023

Grading structure

N	Title	Score	Delivery
1	Report1 : Language	10	02/08/30
2	Report 2 : Evaluation of two PL	15	02/09/10
3	Presentation	10	02/07-09
4	Coding Exercises	15	---
5	Written Exam	25	02/08/…
6	Practical Project	25	02/10/…
7	Research	Unlimited	02/08/20

Github.com/mohammadbtc100

Content at the Glance

- History
- Theories
- Paradigms
- Languages

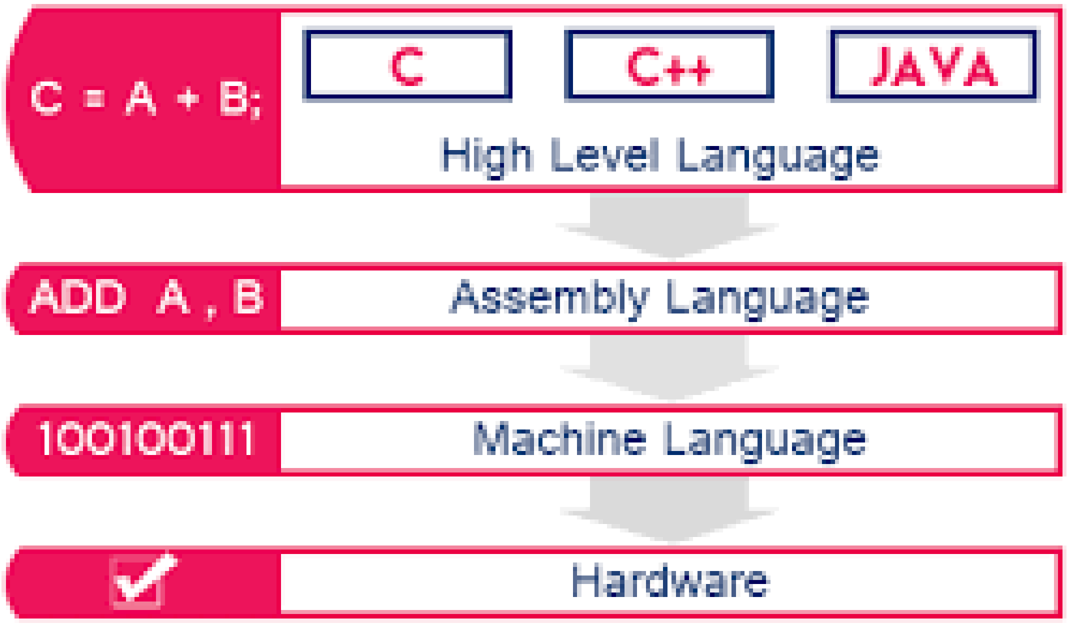
language

FLAMINGTEXT.COM

Chapter 1

History

Code Layers


C = A + B;
C
C++
JAVA
High Level Language
ADD A , B
Assembly Language
100100111
Machine Language
Hardware

History of programming languages

- 1943: Ada Lovelace's machine algorithm Ada Lovelace invents the first-ever machine algorithm for Charles Babbage's Difference Machine that lays the foundation for all programming languages.
- 1944-45: **Plankalkül** Somewhere between 1944-45, Konrad Zuse developed the first 'real' programming language called Plankalkül (Plan Calculus). Zeus's language (among other things) allowed for the creations of procedures, which stored chunks of code that could be invoked over and over to perform routine operations.

Konrad Zuse

Ada Lovelace

History of programming languages

- **1949: Assembly Language** was used in the Electronic Delay Storage Automatic Calculator (EDSAC). Assembly language was a type of low-level programming language that simplified the language of machine code. In other words, the specific instructions necessary to operate a computer.

- **1949: Shortcode** (or Short-order code), was the first High-Level Language (HLL) suggested by John McCauley in 1949. However, it was William Schmitt who implemented it for the BINAC computer the same year and for the UNIVAC in 1950.

- **1952: Autocode** was a general term used for a family of programming languages. First developed by Alick Glennie for the Mark I computer at the University of Manchester, Autocode was the first-ever compiled language to be implemented meaning that it can be translated directly into machine code using a program called a compiler. Autocode was used on the Ferranti Pegasus and Sirius early computing machines in addition to the Mark I.

History of programming languages

- **1957: FORTRAN,** FORmula TRANslation or FORTRAN was created by John Backus and is considered to be the oldest programming language in use today. The programming language was created for high-level Scientific, **Symbolic**, Mathematical, and Statistical computations. FORTRAN is still in use today in some of the world's most advanced supercomputers.

- **1958: LISP** (List Processor)

Originally purposed for artificial intelligence

Invented by John MvCarthy at MIT

Formal notation for Lambda-Calculus => Functional Programming

Pioneered many PL concepts : Garbage collection, Dynamic Types

No distinction between code and data

History of programming languages

- **1958-60: ALGOL 60(Algorithmic Language)**

  Peter Naur

  Great influence on Modern Languages

  Formally specified syntax(BNF)

  Algorithmic View

  Lexical Scoping, Modular Procedures, Recursive Procedures, Variable Type declarations, Stack Storage allocation

  A procedure can be passed to a Procedure

  Recursive call of a Procedure

  ALGOL served as the starting point for the development of some of the most important programming languages including Pascal, C, C++, and Java.

  disadvantages : side-effect, unlimited jump by goto, send parameter only by name

  Algo68 has type conversation for example float to int

  Fortran-> Algol -> Pascal -> Ada -> …

History of programming languages

- **1959:** **COBOL** (Common Business Oriented Language), is the programming language behind many credit card processors, ATMs, telephone and cell calls, hospital signals, and traffic signals systems (just to name a few). The development of the language was led by Dr. Grace Murray Hopper and was designed so that it could run on all brands and types of computers. COBOL is still used to this day primarily for banking and gamification systems.

Sample

English like

stable

Business Oriented

Robust

Easy to Read

Grace Hopper

History of programming languages

- **1960 : Simula**

Ole-Johan Dahl(1931-2002), Kristen Nygaard(1926-2002)

at the Norwegian Computing Center in Oslo

it is an approximate superset of ALGOL 60

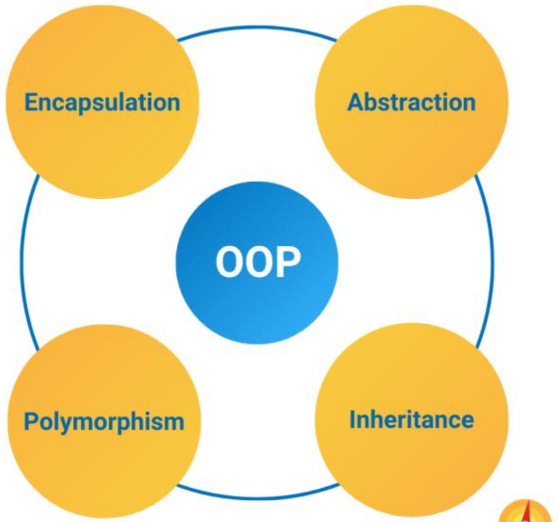
First-Oriented Language

Virtual Procedures

Simula is the name of two simulation programming languages

Simula->Smalltalk

Kristen Nygaard



Ole-Johan Dahl

History of programming languages

- **1964: BASIC** (Beginner's All-Purpose Symbolic Instruction Code) was developed by a group of students at Dartmouth College. The language was written for students who did not have a strong understanding of mathematics or computers. The language was developed further by Microsoft founders Bill Gates and Paul Allen and became the first marketable product of the company.

History of programming languages

- **1970:** **PASCAL** Named after the French mathematician Blaise Pascal, **Niklaus Wirth** developed the programming language in his honor.

Range, Subrange, Variant, and Records

Send parameter by Call by value

And Call by reference

Revised Type System of Algol

It gives the programmer the power to choose the method of passing the parameter.

Algol Like

It was favored by Apple in the company's early days,

because of its ease of use and power.

History of programming languages

- **1972: Smalltalk** developed at the Xerox Palo Alto Research Centre by Alan Kay, Adele Goldberg, and Dan Ingalls, Smalltalk allowed for computer programmers to modify code on the fly. It introduced a variety of programming language aspects that are visible languages of today such as Python, Java, and Ruby. Companies such as Leafly, Logitech, and CrowdStrike state they use Smalltalk in their tech stacks.
- **1972: SQL** (SEQUEL at the time) was first developed by IBM researchers Raymond Boyce and Donald Chamberlain.
- **1978: C** developed by Dennis Ritchie
- **1980/81: Ada** was originally designed by a team led by Jean Ichbiah of CUU Honeywell Bull under contract to the United States Department of Defense. Ada was extended from Pascal
- **1983: C++** Bjarne Stroustrup modified the C language at the Bell Labs

History of programming languages

- **1983: Objective-C** developed by Brad Cox and Tom Love, Objective-C is the main programming language used to write software for macOS and iOS, Apple's operating systems.
- **1987: Perl** was created by Larry Wall and is a general-purpose, high-level programming language. It designed for text editing.
- **1990: Haskell** is a general-purpose programming language named after the American logician and mathematician Haskell Brooks Curry. It is a purely functional.
- **1991: Python** named after the British comedy troupe 'Monty Python', Python was developed by Guido Van Rossum. It is a general-purpose, high-level programming language.
- **1991: Visual Basic** developed by Microsoft, Visual Basic allows programmers to utilize a drag-and-drop style of choosing and changing pre-selected chunks of code through a graphical user interface (GUI)
- **1993: Ruby** created by Yukihiro Matsumoto, Ruby is an interpreted high-level programming language. A teaching language which was influenced by Perl, Ada, Lisp, and Smalltalk.

History of programming languages

- ❑ **1995:** **Java** is a general-purpose, high-level language created by James Gosling for an interactive TV project. It has cross-platform functionality.

Originally called Oak

Interface, Single Inheritance, Exception handling, built-in threading model, references & automatic garbage collection

no multiple inheritance, no operator overloading

Java=C++ + Modula-3

JSP(Java Server Page) is used for web development in Java language.

Framework :

- ❑ Spring
- ❑ Hibernate
- ❑ Strut

Application:

- ❑ Web, Mobile and Desktop
- ❑ Enterprise
- ❑ Financial

History of programming languages

- **1995: PHP,** Formerly known as 'Personal Home Page' which now stands for 'Hypertext Preprocessor', PHP was developed by Rasmus Lerdorf. Its primary uses include building and maintaining dynamic web pages, as well as server-side development.
- **1995: JavaScript,** was created by Brendan Eich.
- **2001: C#** developed at Microsoft with the hope of combining the computing ability of C++ with the simplicity of Visual Basic.
- **2003: Scala** developed by Martin Odersky, Scala which combines mathematical functional programming and organized object-oriented programming on Java Virtual Machine.
- **2003: Groovy** derived from Java, Groovy was developed by James Strachan and Bob McWhirter. The language improves productivity because of its succinct and easy to learn.
- **2009: Go** developed by Google
- **2014: Swift** developed by Apple as a replacement for C, C++, and Objective-C
- **2015 : Rust** developed by Mozilla

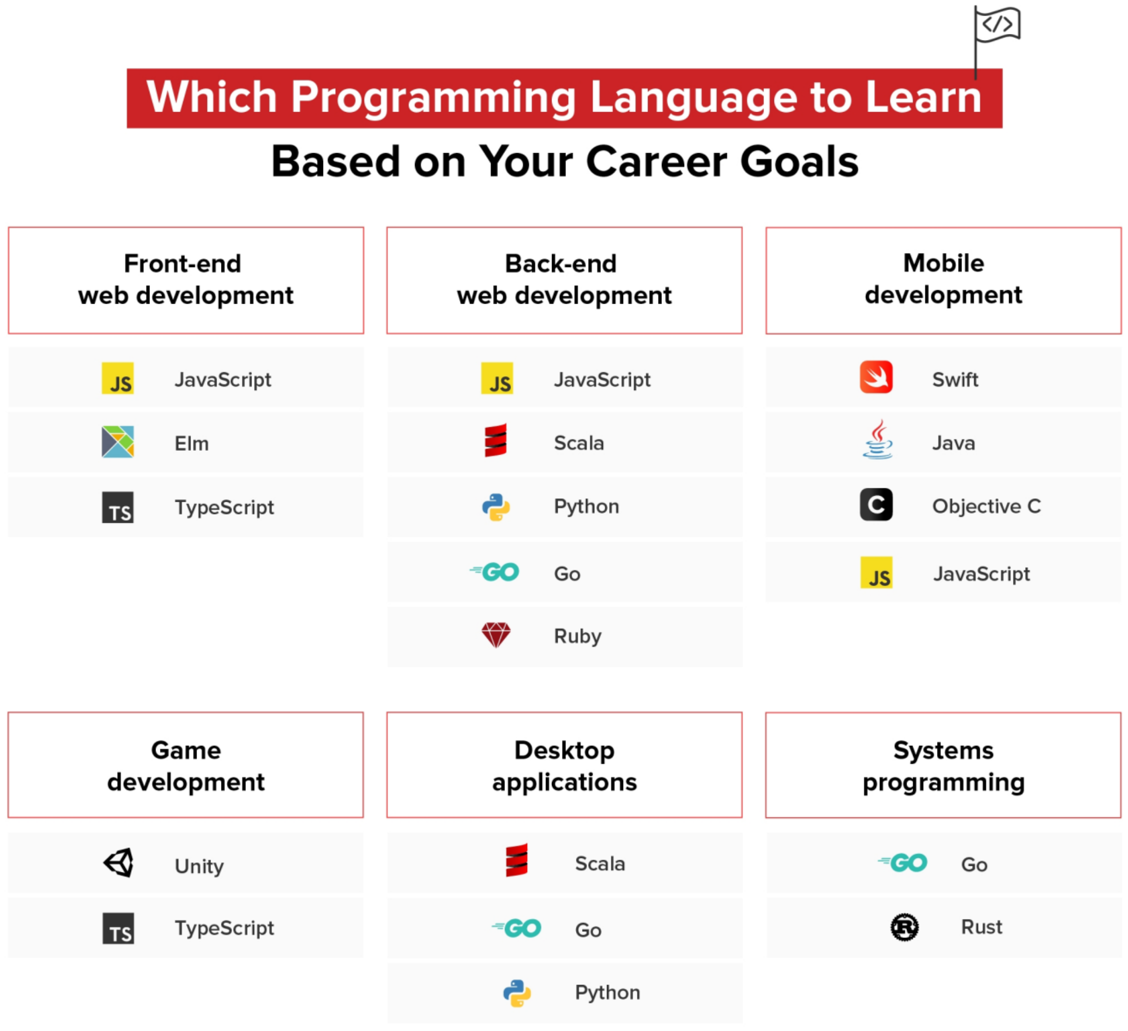
Top Languages -> TIOBE Index for September 2023

2023	2022	Change	Programming Language	Ratings	Change
1	1		Python	14.16%	-1.58%
2	2		C	11.27%	-2.70%
3	4	▲	C++	10.65%	+0.90%
4	3	▼	Java	9.49%	-2.23%
5	5		C#	7.31%	+2.42%
6	7	▲	JavaScript	3.30%	+0.48%
7	6	▼	Visual Basic	2.22%	-2.18%
8	10	▲	PHP	1.55%	-0.13%
9	8	▼	Assembly language	1.53%	-0.96%
10	9	▼	SQL	1.44%	-0.57%
11	15	▲▲	Fortran	1.28%	+0.26%
12	12		Go	1.19%	+0.03%
13	14	▲	MATLAB	1.19%	+0.13%
14	22	▲▲	Scratch	1.08%	+0.51%
15	13	▼	Delphi/Object Pascal	1.02%	-0.07%
16	16		Swift	1.00%	+0.02%
17	26	▲▲	Rust	0.97%	+0.47%
18	18		R	0.97%	+0.02%
19	20	▲	Ruby	0.95%	+0.30%
20	34	▲▲	Kotlin	0.90%	+0.59%

Programming Language Hall of Fame

Year	Winner
2022	C++
2021	Python
2020	Python
2019	C
2018	Python
2017	C
2016	Go
2015	Java
2014	JavaScript
2013	Transact-SQL
2012	Objective-C
2011	Objective-C
2010	Python
2009	Go
2008	C
2007	Python
2006	Ruby
2005	Java
2004	PHP
2003	C++

Applications



Which Programming Language to Learn

Based on Your Career Goals

Front-end web development
- JavaScript
- Elm
- TypeScript

Back-end web development
- JavaScript
- Scala
- Python
- Go
- Ruby

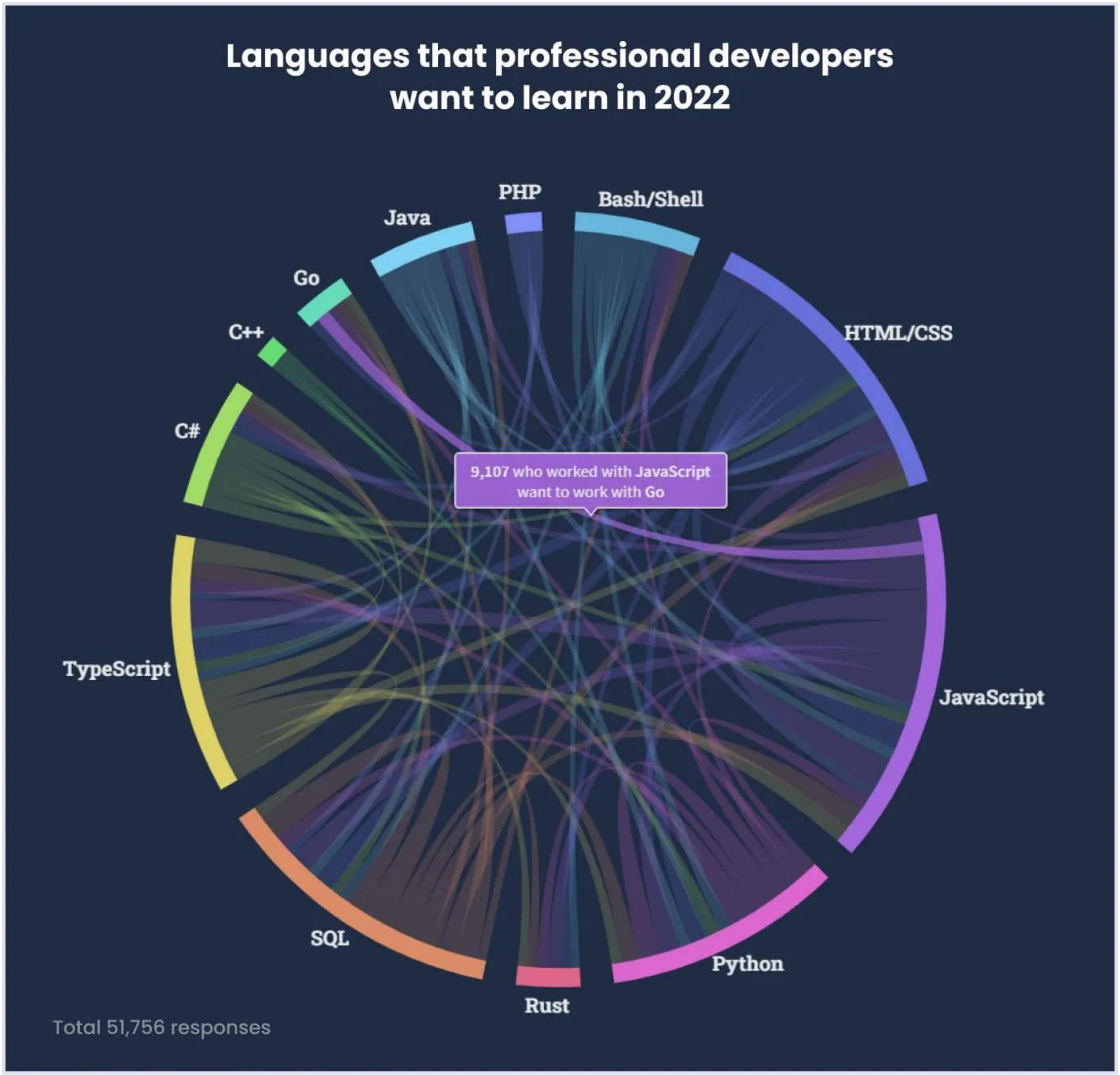
Mobile development
- Swift
- Java
- Objective C
- JavaScript

Game development
- Unity
- TypeScript

Desktop applications
- Scala
- Go
- Python

Systems programming
- Go
- Rust

Languages that Professional developers want to learn

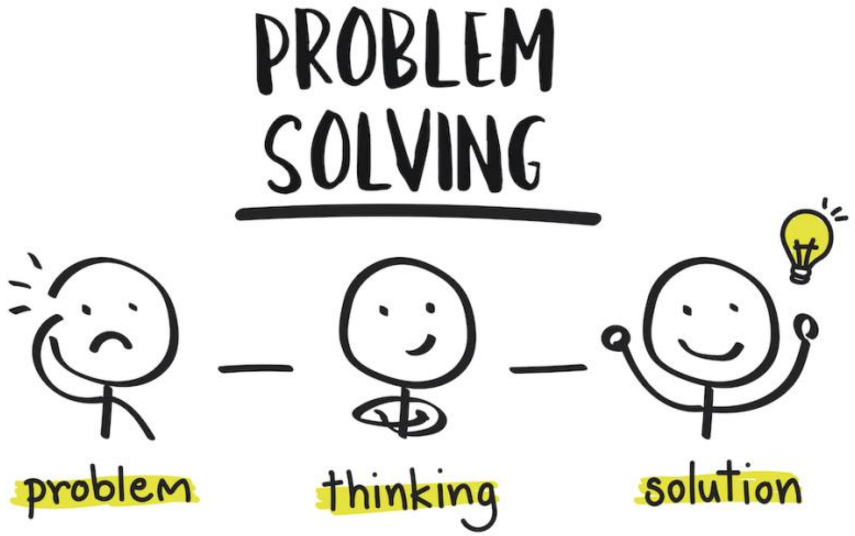


Chapter 2

Theories

Main Theories

N	Theory	Paradigms
1	Predicate logic	Logic Programming
2	Turing Machine	Imperative Programming
3	Lambda Calculus	Functional Programming
4	Automata Theory	Recursive Functions & Automata, Regex

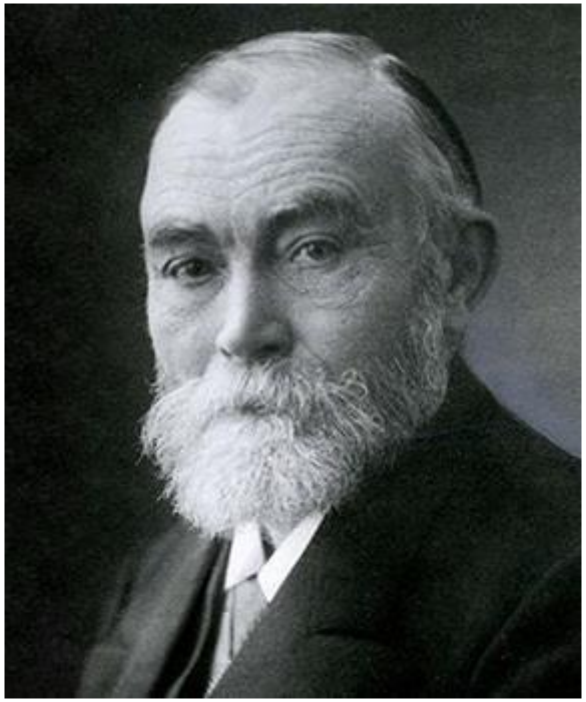


- ❑ Machine Theory and Automata
- ❑ Compiler Design
- ❑ Programming Language Design

λ

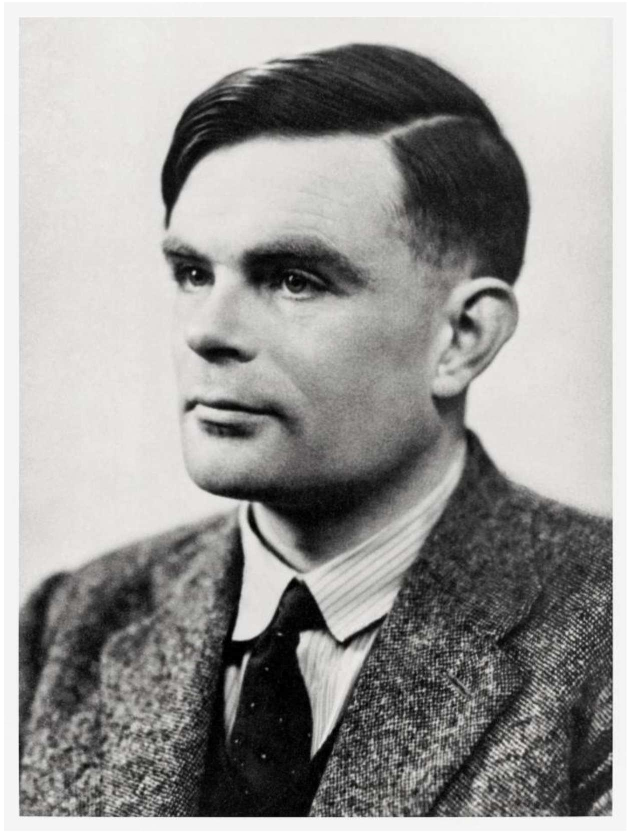
Predicate logic

- Formal Basis for Proof Theory and automated Theorem proving
- **Gottlob Frege**, Logition, Germany(1848-1925)
- Mathmatics Logic field
- **Logic Programming :** Computation as Logical deduction
- Example : **Prolog**



Turing Machines

- **Alen Turing(1912-1954)**
- **Imperative Programming** : Sequences of commands, explicit state transitions, update via assignment
- Example : **Lisp**



Lambda Calculus

- **Alonzo Church(1903-1995)**
- Formal basis for all functional languages, Semantics,

  Type theory

- **Functional Programming** : Pure expression evaluation,

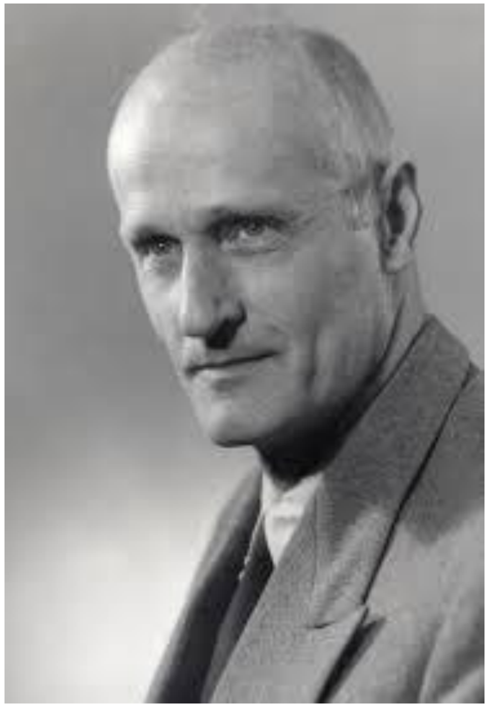
  No assignment operator
- Examples : **Haskell**, **Groovy**

Functional Languages:

- It is easier to describe the complex program.
- Portability and Resolvability are more important than Efficiency
- For this reason, they are more Interpretive and on a Virtual Machine.
- They are generally High Level

Automata Theory

- **Recursive Functions & Automata**
- **Stephen Kleene(1909-1994)**
- Regular expressions
- finite-state machines
- PDA(pushdown automaton)



Others

Combinatory Logic

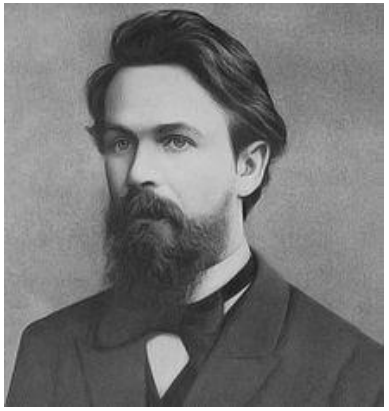
- Moses Schonfinkel, Logition(1889-1942)
- Haskell Curry(1900-1982)
- Effective in the development of functional languages

Post Production Systems

- Emil Post(1897-1954)

Markov Algorithms

- Andrey Markov(1903-1979)

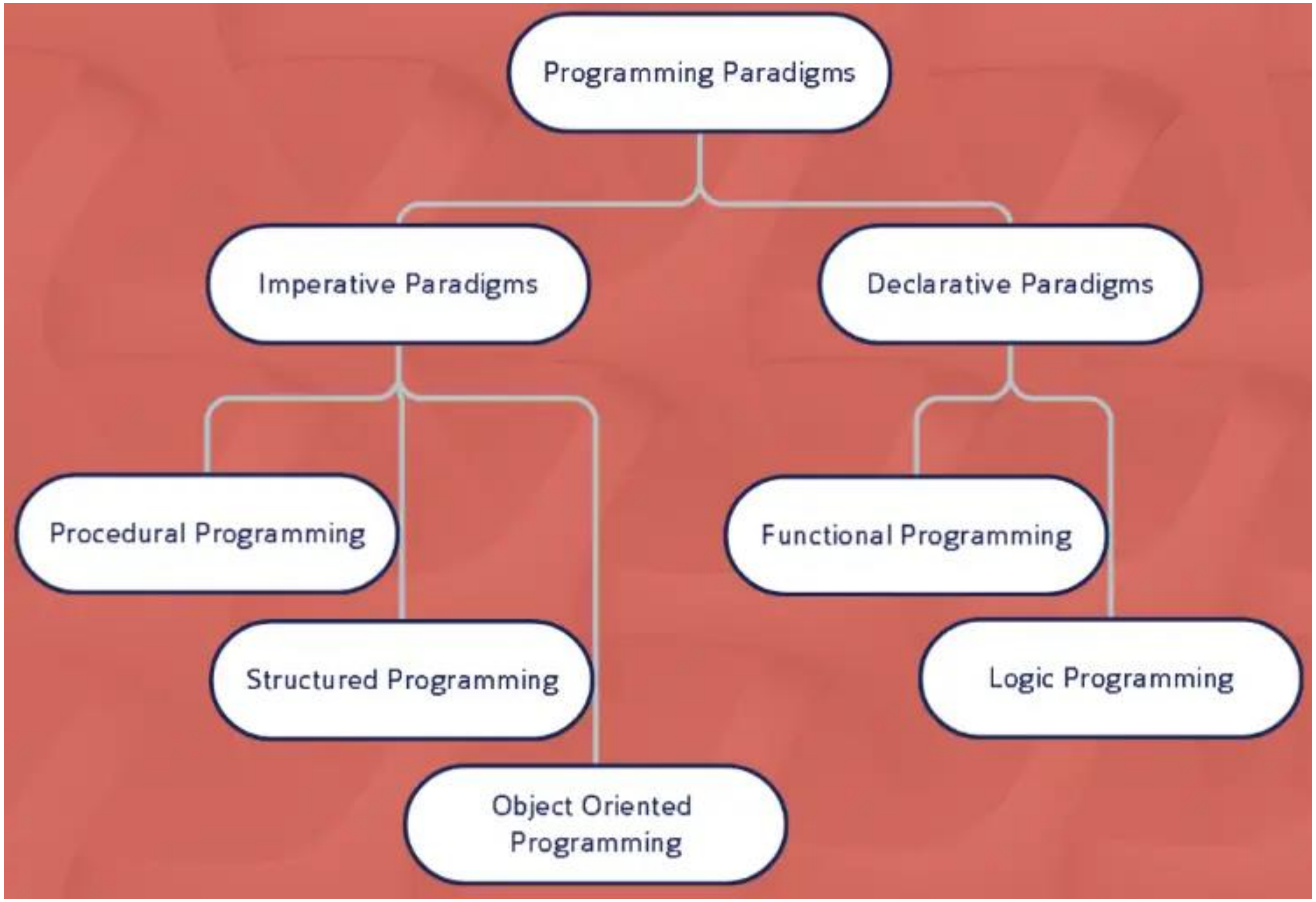


Chapter 3

Paradigms

Paradigms

Main Paradigms :



Paradigms

Imperative Programming

Imperative programming is a programming paradigm that uses statements to change a program's state. It's concerned with describing how a program operates in a step-by-step manner.

Statements are executed, and the results are stored in variables. It is more of a line-by-line instruction given to the computer.

Let us understand imperative programming using a C++ program that gives us factorial of a number.

Imperative programming can be further divided into procedural, structured, and object-oriented programming.

Paradigms

Procedural Programming :

Procedural programming can also be referred to as imperative programming. It is a programming paradigm based upon the concept of procedure calls, in which statements are structured into procedures (also known as subroutines or functions). They are a list of instructions to tell the computer what to do step by step, Procedural programming languages are known as top-down languages. Most of the early programming languages are all procedural.

Features:

- ❑ Procedural Programming is excellent for general-purpose programming
- ❑ The coded simplicity along with ease of implementation of compilers and interpreters
- ❑ A large variety of books and online course material available on tested algorithms, making it easier to learn.
- ❑ The source code is portable
- ❑ The code can be reused in different parts of the program, without the need to copy it
- ❑ The program flow can be tracked easily as it has a top-down approach.

Examples of Fortran **C** and Cobol

Paradigms

Structured Programming :

It is an imperative programming paradigm where we use nested loops, conditional statements, and subroutines to define the control flow of a program. This paradigm has a top-down implementation which promotes code readability and reusability. In commercial applications, this helps to reduce development time and increase code quality.

Structured programming uses control structures like loops (for, while), conditionals (if, switch), and subroutines to control the program's flow. These structures enable programmers to write code that is easier to understand and modify, because they provide a clear, top-down control flow.

While structured programming improves the readability and organization of code, it can lead to more complex code structures in larger programs. It may also require more overhead in terms of planning and design compared to less structured paradigms. However, this doesn't necessarily mean that structured programming languages are less efficient in execution. The efficiency often depends on the implementation details and the specific problem being addressed.

Some of the languages that use Structured programming paradigms are:

- ❑ Algol 60
- ❑ PL/I
- ❑ Pascal
- ❑ Ada 83
- ❑ Modula

Paradigms

Object-Oriented Programming(OO):

In this framework, all real-world entities are represented by Classes. Objects are instances of classes so each object encapsulates a state and behavior. State implies the fields, attributes of the object and behavior is what you do with the state of the object and they are the methods. Objects interact with each other by passing messages.

Features :

- Encapsulation : It is the binding of data into a single unit.
- Inheritance : It helps in establishing hierarchical relationships promoting code reusability.
- Abstraction : It helps in hiding unnecessary attributes while showing the ones required.
- Polymorphism : It allows different objects to respond in different ways to the same input.

Programming languages that have implemented the OO paradigm are: Ruby, **Java**, **C++, C#**, **Python**, Simula (the first OOP language)

Paradigms

Declarative Programming :

Declarative programming is a programming approach that expresses a computation's logic without discussing its control flow. The programmer must specify what the program must accomplish but need not specify how it must be implemented.

In other words, rather than dictating 'how' to achieve a goal, as is common in imperative programming, declarative programming focuses on 'what' needs to be achieved, leaving the specifics of 'how' to the underlying system or language implementation.

While you don't have to specify 'how' to achieve something, the 'how' is still important – it's just handled by the system or the language implementation, not the programmer.

The declarative programming paradigm is further divided into logic programming and functional programming.

Paradigms

Logical Programming:

Logical programming is a computer programming paradigm that has its foundations in mathematical logic in which program statements express facts and rules about problems within a system. Rules are written as logical clauses with a head and a body. They also follow a declarative rather than an imperative approach. However, what does that mean?
To understand how a problem can be solved in logical programming, you need to know about the building blocks − Facts and Rules − **like** **Prolog, ACL2, Isabelle and HOL**

There are three basic statements:

- ❑ **Facts** are fundamental assertions about the problem domain (e.g. "Moses is a man")
- ❑ **Rules** are inferences about facts in the domain (e.g. "All men are mortal.")
- ❑ **Queries** are questions about that domain (e.g. "Is Moses mortal?")

Features:

- ❑ Logical programming can be used to express knowledge in a way that does not depend on the implementation, making programs more flexible, compressed and understandable.
- ❑ It enables knowledge to be separated from use, i.e. the machine architecture can be changed without changing programs or their underlying code.
- ❑ It can be altered and extended in natural ways to support special forms of knowledge, such as meta-level of higher-order knowledge.
- ❑ It can be used in non-computational disciplines relying on reasoning and precise means of expression.

Paradigms

Functional Programming :

Functional programming is a programming paradigm where you have a style of building the structure and elements of computer programs. Here you treat computation as an evaluation of mathematical functions and you avoid changing-state and mutable data.

Functional programming consists only of PURE functions. So, what do you understand by Pure functions?
Pure functions are those which take an argument list as an input and whose output is a return value. Now you may feel that all functions are pure as any function takes in values and returns a value.

Features :

- ❑ Pure functions
- ❑ Recursion
- ❑ Referential transparency
- ❑ Functions are First-Class and can be Higher-Order
- ❑ Variables are Immutable

Programming Languages that support functional programming: **Haskell**, JavaScript, Scala, **Erlang**, **Elixir**, Lisp, ML, **Clojure**, **OCaml**, F#, Common Lisp, Racket.

Paradigms at the Glance

Imperative: Programming with an explicit sequence of commands that update state.

Declarative: Programming by specifying the result you want, not how to get it.

Structured: Programming with clean, goto-free, nested control structures.

Procedural: Imperative programming with procedure calls.

Functional (Applicative): Programming with function calls that avoid any global state.

Function-Level (Combinator): Programming with no variables at all.

Object-Oriented: Programming by defining objects that send messages to each other. Objects have their own internal (encapsulated) state and public interfaces. Object orientation can be:

- **Class-based**: Objects get state and behavior based on membership in a class.
- **Prototype-based**: Objects get behavior from a prototype object.

Event-Driven: Programming with emitters and listeners of asynchronous actions.

Aspect-Oriented: Programming cross-cutting concerns applied transparently.

Reflective: Programming by manipulating the program elements themselves.

Array: Programming with powerful array operators that usually make loops unnecessary.

Most Important Programming Language

N	Title	Date	Imples	Style	Domain
1	Fortran	1957	------	Structured	Modeling, Calc
2	Cobol	1959	------	Proc, OOP	Business
3	VB	1991-1998	------	Object-Based	Business
4	FoxPro	1991-2015	------	Proc, OOP	DBMS
5	Lua	1993	C	Imperative	System, Game
6	Erlang	1986	Erlang/C	Concurrent, Functional	Infrastructure
7	Haskell	1990	Haskell/C	Functional	Research, Industry, Enterprise App
8	oCaml	1996	oCaml/C	Functional,OOP	Sys,Web,Fin
9	Perl	1987	C	Functional, OOP	Text Processing

MP : Multi-Paradigm, Calc : Calculation, OOP : Object-Oriented Programming

Most Important Programming Language

N	Title	Date	Implemented	Style	Domain
10	JS/TS	1995	------	Event-Driven	Web
11	Python	1991	C	OOP, Func	AI, Startup
12	C/C++	1978	C-Ada/C++	Structured, Proc	System
13	Java	1995	----	OOP	Financial
14	C#	2001	----	OOP	Business/Org
15	Go	2011	Go	Concurrent, OOP	Backend/IO
16	R	1993	C/Fortran	Proc	Data Analysis
17	PHP	1995	C/C++	Imperative	Web-Backend
18	Rust	2015	C/C++	Concurrent, Func	System
19	Solidity	2014	C++	OOP	Blockchain
20	Clojure				
21	Julia				

MP : Multi-Paradigm, Proc : Procedural, Func : Functional

Chapter 4

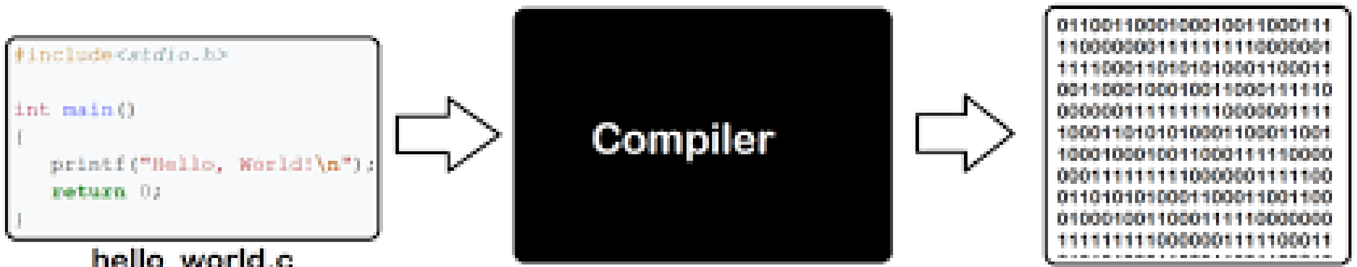
Languages

Translate Code

- ❑ Source Code
- ❑ Byte-Code
- ❑ Native Code
- ❑ Virtual Machine
- ❑ Interpreter
- ❑ Compiler
- ❑ Just-In-Time Compiler(JIT)

COMPILER

1-A **Compiler** is a special program that translates a programming language's source code into machine code, bytecode or another programming language



```c
#include<stdio.h>

int main()
{
   printf("Hello, World!\n");
   return 0;
}
```

hello_world.c

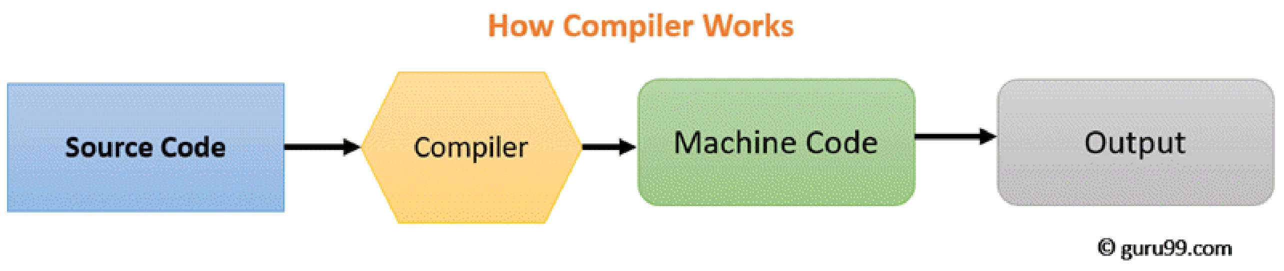
```
01100110001000100110000111
11000000011111111110000001
11110001101010100011000111
00110001000100110001111110
00000011111111110000001111
10001101010100011000110001
10001000100110001111110000
00011111111110000001111100
01101010100011000110011000
01000100110001111110000000
11111111100000011111100011
```

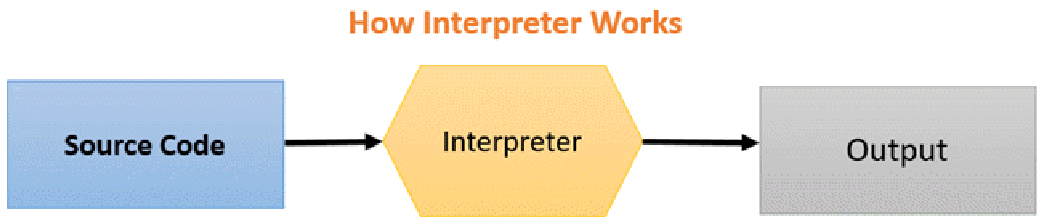
hello_world.o

Translate Code

An **Interpreter** is a computer program that directly executes instructions written in a programming or scripting language, without requiring them previously to have been compiled into a machine language program. An interpreter generally uses one of the following strategies for program execution:

- ❑ Parse the source code and perform its behavior directly;
- ❑ Translate source code into some efficient intermediate representation or object code and immediately execute that;
- ❑ Explicitly execute stored precompiled bytecode made by a compiler and matched with the interpreter Virtual Machine.

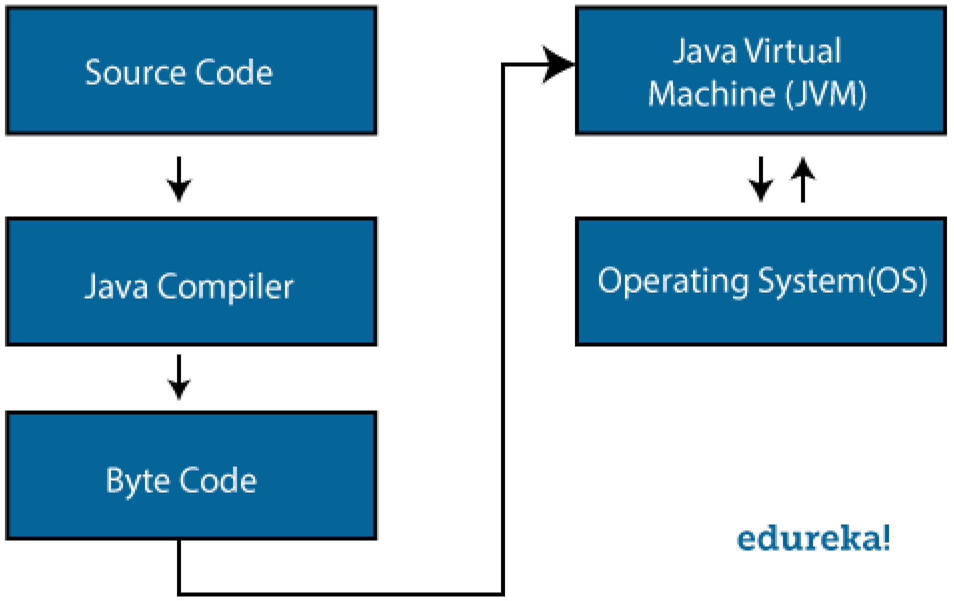




Translate Code

Virtual Machine or Language Runtime System

For example Java Virtual Machine(JVM)



Languages studied

N	Language	Desired Paradigm	N of Session
1	Review of Assembly Language	Structured	1
2	C	Procedural	1
3	C++	Object Oriented	1
4	Rust	Functional	1
5	Go	Concurrent	3
6	Java Script	Event-Driven	4
7	Solidity	Object Oriented	1

Paradigms

A programming paradigm is a style, or “way,” of programming.

N	Paradigm	Languages
1	Procedural	C, JavaScript
2	Functional	Rust, JavaScript
3	Object-Oriented	C++, JavaScript
4	Concurrent	Rust, Go
5	Event-Driven	JavaScript
6	Imperative	Rust, Go, JavaScript
7	Generic	Rust, Go, C++
8	and etc…	

Issues

- Metaprogramming : It is writing programs that operate on other programs
- **Reflection** : Reflection is the ability of a program to introspect and analyze its structure during run-time.
- Concurrency
- Classes
- Object Orientation
- Functional Programming
- Subroutines
- Expression Evaluation
- Binding
- OHM Library
- Language Theory
- **Context Free Grammar**

C Language and Procedural Paradigm

C is a general-purpose computer 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, device drivers, 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.

Paradigm : Multi-paradigm: imperative (**procedural**), structured

Designed by Dennis Ritchie, Ken Thompson

Major implementations : GCC, Clang

Every action is Function

Anything that is 0 is true and anything that is not 0 is true

Early C++, Modula-3, Eiffel source-translated to C

C++ Language and Object Oriented Paradigm

C and C++ are often referred to as low-level languages. They are especially useful for building high-performance systems. C++ can be thought of as an extension to the C language, adding features like object-oriented programming. A few projects that use these languages include the Linux Kernel, which is written in C, and Adobe Photoshop, which is written in C++.

C and C++ may seem intimidating because of the added complexity of the way you manage and interact with the programs' memory. However, learning C and C++ can provide you with a strong foundation in programming, making it easy to understand more nuanced concepts.

C++ = C + Simula

C++ Features:

- ❑ Multiple Inheritance
- ❑ Templates/Generics
- ❑ Exception Handling

C Framework :

- ❑ Unlimited…

C Application:

- ❑ Everywhere

C++ Framework :

- ❑ Qt
- ❑ ROOT
- ❑ Ffead-Cpp
- ❑ Boost

C++ Application:

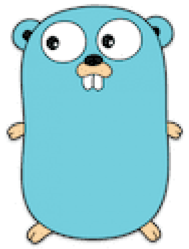
- ❑ Everywhere

Rust Language and Functional Paradigm

Rust is another general-purpose programming language based on C and C++. Rust is strict on how you interact with memory. This approach minimizes the possibility of bugs or vulnerabilities in programs while maintaining a high level of performance. As a result, Rust is an increasingly popular choice for building systems where safety and security are essential.

One great demonstration of how Rust is used can be found in the Firefox browser itself. It was used to improve the efficiency of the CSS engine while simultaneously reducing the number of potential security vulnerabilities. Even though Rust is especially appropriate for system-level programming, it’s not unheard of for people to start their programming journey with Rust.

Golang Language and Concurrent Paradigm



Go, also known as Golang, is a general-purpose programming language developed by Google with syntax and use cases similar to C and C++. Its goals were to simplify the syntax and general complexity of these and other languages. Go can be used in systems engineering, software engineering, and data science.

Go is often used to implement components of larger projects where execution speed is important. For example, in 2014, Dropbox used Go to improve the performance of its backend systems, especially concerning access to its database

Framework :
- ❑ Gin
- ❑ Revel
- ❑ Beego

Application:
- ❑ Backend

JavaScript Language and Event-Driven Paradigm

JavaScript is commonly used with HTML and CSS to implement client-side functionality using small scripts. This is why it's often called a scripting language. You can use JavaScript to add features like dropdown menus and web applications to your website. It is beginner-friendly in that it doesn't need a compiler to run, most browsers can serve as the default environment.

Beyond that, you can utilize frameworks like Node.js to use JavaScript in server-side programming. It is powerful enough to accomplish tasks like implementing a search algorithm, analyzing data provided by a user, or solving mathematical equations. There are even game engines built with JavaScript that can have impressive results.

Framework :

- ❑ Node.js
- ❑ Angular, React.js and Vue js
- ❑ Meteor and Expo

Application:

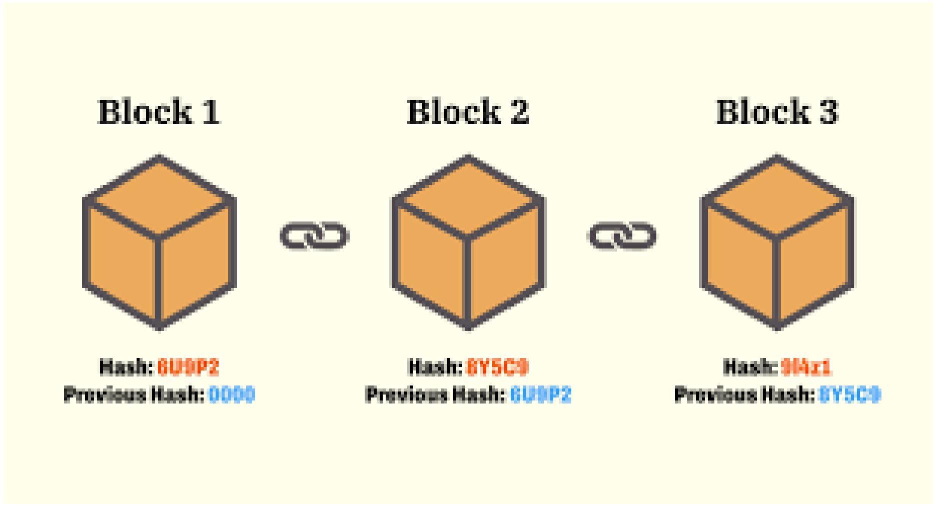
- ❑ Web
- ❑ Mobile
- ❑ Desktop

Blockchain and Smart Contract

A Blockchain is a decentralized, distributed and public digital ledger that is used to record transactions across many computers so that the record cannot be altered retroactively without the alteration of all subsequent blocks and the consensus of the network.

Blockchain Languages :

- ❑ C/C++
- ❑ JS/TS
- ❑ Golang
- ❑ Rust



Smart Contract Languages :

- ❑ Solidity
- ❑ Vyper
- ❑ Also all above languages

* Java family get great position in Blockchain field near soon

Solidity

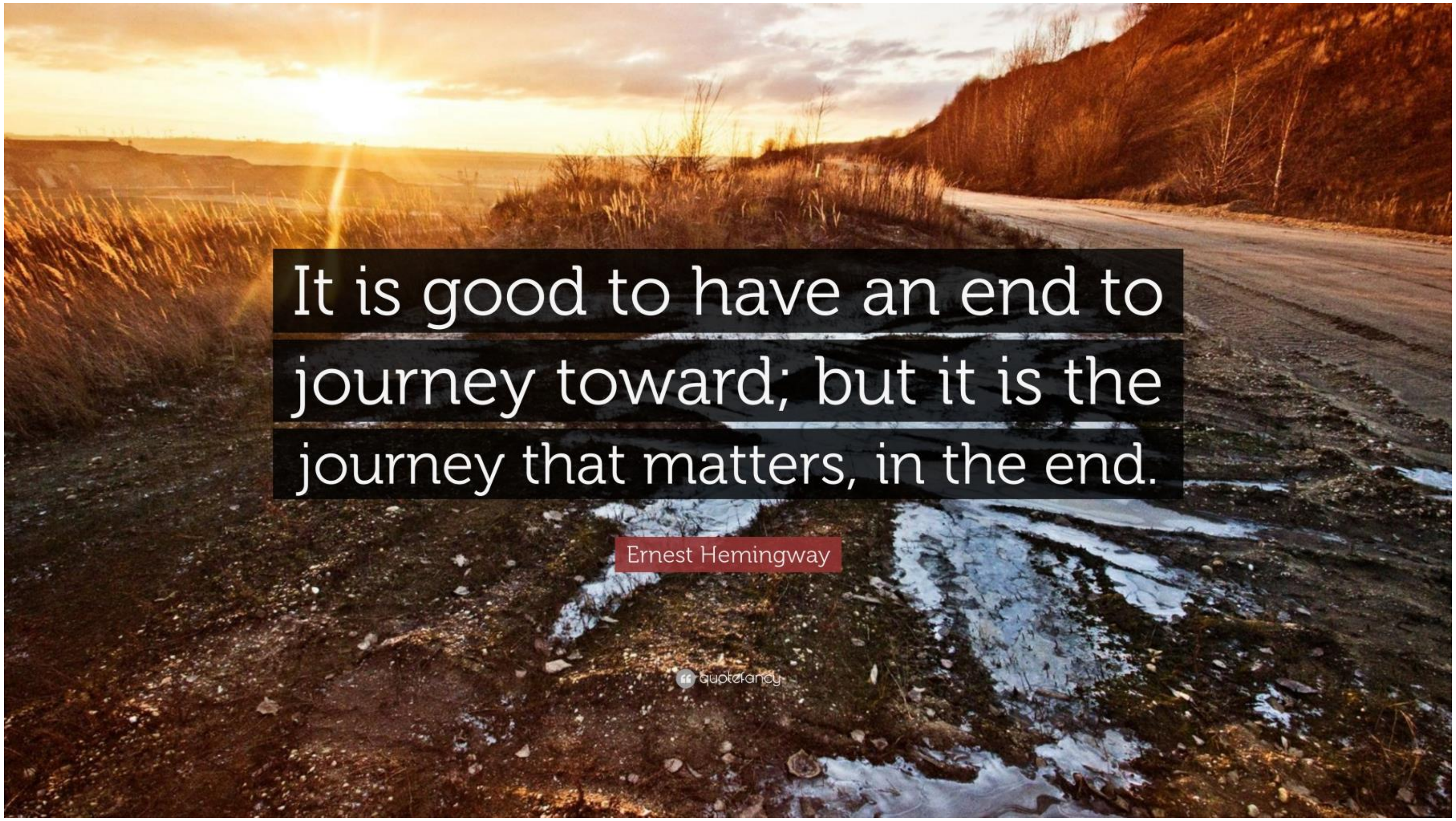
Solidity is an object-oriented programming language for implementing smart contracts on various Blockchain platforms, most notably, Ethereum.

Designed by: Gavin Wood
Developer: Christian Reitwiessner, Alex Beregszaszi
First appeared: August 2014
Implementation language: C++
Paradigm: Object-oriented

Exercises

1) Chat program between two computers
2) Client and Multi-threaded Server Program for send a file

The end of the journey


It is good to have an end to journey toward; but it is the journey that matters, in the end.
Ernest Hemingway
quotefancy

References

References

- ❑ **Programming paradigm**

https://en.wikipedia.org/wiki/Programming_paradigm
https://cs.lmu.edu/~ray/notes/paradigms/

- ❑ **Functional Paradigm**

https://www.geeksforgeeks.org/functional-programming-paradigm/

- ❑ **C++ Language**

https://www.cs.mtsu.edu/~xyang/2170/

- ❑ **Reactjs**

https://reactapp.ir/

- ❑ **Programming language theory**

https://en.wikipedia.org/wiki/Programming_language_theory
https://academic-accelerator.com/encyclopedia/programming-language-theory
https://cs.lmu.edu/~ray/notes/languagetheory/

- ❑ **Language Theories**

https://cs.lmu.edu/~ray/notes/languagetheory/
https://en.wikipedia.org/wiki/Programming_language_theory

- ❑ **Others**

https://cs.lmu.edu/~ray/
https://cs.lmu.edu/~ray/classes/pls/
https://careerkarma.com/blog/easiest-programming-languages-to-learn/

References

- ❑ **Others**

https://blog.faradars.org
https://hackr.io/blog/programming-paradigms
https://www.cs.cornell.edu/courses/cs4110/2012fa/

- ❑ **System Programming**

https://www.systutorials.com/hashing-library-for-c/

Appendices

Appendices

Comparison of two Languages

N	Title	Student
1	Python vs Julia	Parsa Arianpoor
2	PHP vs Ruby	Abdorahman Bigonahi
3	PowerShell vs Bash	Aliakbar Makarian
4	C# vs Java	Sina Abbasi, Alireza Sharifi
5	Kotlin	Mohammadreza HaydariKia
6	Objective-C vs Swift	
7	Erlang vs Elixir	Mohammad Farazendeh, Omidreza Sanjari
8	Lua vs Perl	S. Mickaeil Sediqe, Ali Qanbari
9	Haskell	Fardin Sedaqat
10	oCaml	
11	Elm vs Dart	Amin Khoshi, Abbas Kargar Jahromi
12	Golang	Mahdi Salari
13	Scala	Mohammadhadi Jedabi

*Students must write one or two report on the language presented.

Report of a Languages

N	Students	Languages
1	Dart	Alireza Naseri****
2	Rust	Mohammadsadeq Mohebi
3	Golang	Mohammadamin Naseri
4	Groovy	
5	Solidity	M.Hossein Khabiri
6	F#	Ali Hajihossini
7	C++	Mohammad Salehi
8	C	Yasin Shahriari
9		
10		
11		
12		
13		

Mini Project

- ❑ CRUD on JSON file
- ❑ Client/Server chat system
- ❑ Upload file on system
- ❑ CRUD on Database by RESTful API
- ❑ Hash and Encryption Data by RESTful API
- ❑ Sorting Algorithm by RESTful API
- ❑ Searching Algorithms by RESTful API
- ❑ Store Linked-List in JSON file
- ❑ Sample Email System behind a RESTful API

Tools

- Postman
- Jira
- Git, Github and GitLab
- Chrome or Firefox DevTools
- Stack Overflow
- Docker

1-HTML

HTML is a markup language used in web development and is said to be one of the easiest coding languages to learn for newbies.

HTML stands for Hypertext Markup Language. It is one of the three basic building blocks of the web. You might recognize "HTML" as part of a link to a website or in your browser's address bar. Although it isn't technically a programming language, learning how to create an HTML page is often the first step in expanding your coding knowledge.

HTML is commonly used alongside JavaScript and CSS for front end web development. HTML is great for beginners because it has a simple structure with easy-to-define opening and closing tags. You also use elements in HTML to decide how text, images, and other interactive buttons are shown on a web page.

2-CSS

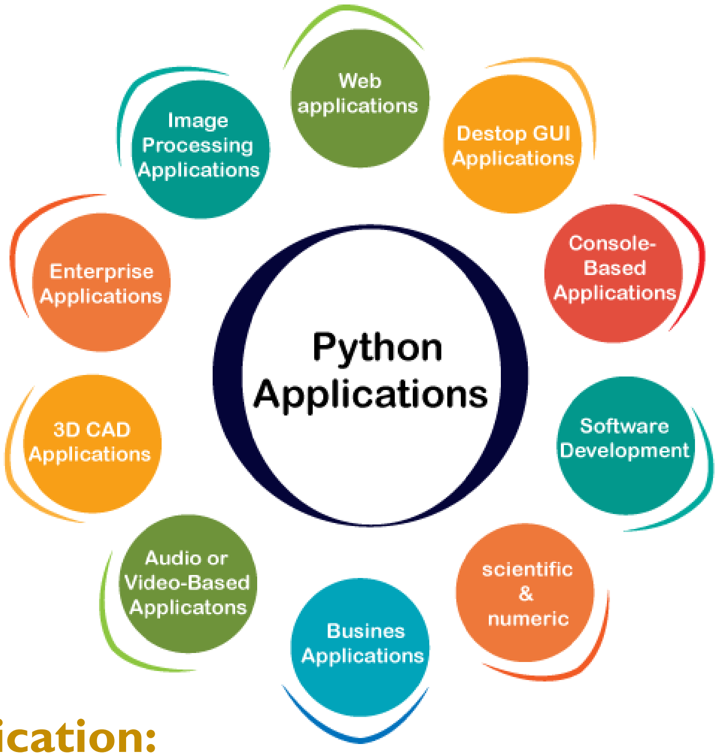
Cascading Style Sheets (CSS) is designed to be a simple coding language. It is used to customize the appearance of a website and define how it should look when viewed in a browser. You can use CSS to define animations, hide or show certain content, and control the positioning of the page's elements. CSS can improve the look, readability, and functionality of a website.

In fact, the article you're reading right now is built with HTML, CSS, and JavaScript. CSS is an ideal language for beginners because, like HTML, it will help you execute basic development tasks.

4-Python

Python is a general-purpose coding language designed with readability in mind. As one of the easiest programming languages, it's often used as an introductory language for computer science students in college. Python is very versatile and can be used to develop applications, automate processes, build operating systems, and more.

Python makes an excellent choice for beginners because it uses a simple syntax, comparable to the English language. This can make coding less intimidating for beginners. Python is also supported by a huge community of developers with lots of resources available to help beginners find their footing.



Framework :

- ❑ Django
- ❑ Flask

Application:

- ❑ Web
- ❑ AI

5-Ruby

Ruby is a powerful programming language with a focus on simplicity and productivity. Ruby is often compared to Python, as the two languages have common use cases and similar features. With Ruby, you can perform complex tasks using minimal code. Companies like Airbnb, Hulu, and Shopify reportedly use Ruby in their tech stacks for web development.

It is also an open-source programming language which means beginners have access to plenty of resources like tutorials, books on Ruby programming, and online discussions. Ruby is commonly used with its framework Rails to create dynamic and responsive web applications.

7-PHP

PHP, or Hypertext Preprocessor, is one of the easiest programming languages to learn, especially if you already know HTML. It is mainly used to provide the backend server functionality that is essential to many websites today. PHP makes it easy for developers to retrieve and store data in databases while also processing and replying to users' requests.

Creating your first PHP program is as easy as embedding PHP code into an HTML page and uploading the file to a server that can process it. Many online resources can help you learn PHP. Some notable projects and services that use PHP include Facebook, Tumblr, and WordPress.

Framework :

- ❑ CakePHP
- ❑ Laravel
- ❑ Symfony
- ❑ Phalcon

Application:

- ❑ Web

9-C#

C# is a general-purpose coding language that is developed by Microsoft for Microsoft. It can be used to develop web apps, cloud-based services, enterprise software, and more. It also provides full support for object-oriented programming. Perhaps one of the most recognizable use cases of C# is game development using the popular Unity game engine.

Framework :

- ❑ Dot NET
- ❑ Xamarin

Application:

- ❑ Web, Mobile and Desktop

10- R

R is a coding language that is primarily used for data science, deep learning, and machine learning. If you're just starting with programming, you may find that R has a steep learning curve. However, there are a lot of R add-on tools like RStudio and lintr that can help simplify the process.

Like Python, R is commonly used to perform statistical analysis of data, build web applications, and write scripts to accomplish various tasks. Learning R is a great option if you're considering becoming a data scientist or simply love working with statistics.

Application:

- ❑ Data Analysais

13-Swift

Apple developed Swift in 2014 specifically for use within its own technological ecosystems like macOS and iOS. Swift is also easy to read and write, so it's a good option for novice programmers. To guide beginning coders through the Swift programming language, Apple created a free application called Swift Playgrounds.

Even if you have zero programming experience, Swift Playgrounds can help you quickly learn the fundamentals of Swift before moving on to more complex concepts. Swift Playgrounds provides direct guidance as you learn Swift, and it's a useful resource that will teach you skills applicable to any programming field.

Framework :

- ❑ Alamofire
- ❑ RxSwift
- ❑ Snapkit

Application:

- ❑ Apple Products
- ❑ Web, Mobile and Desktop

14-Erlang

Erlang is a functional, general-purpose programming language. As one of the easiest programming languages for beginners, it made its first appearance in 1986 with a focus on concurrent programming. That means its features are ideal for building systems where multiple, distinct processes need to run simultaneously while still communicating with each other.
Unlike other programming languages on this list, Erlang only supports functional programming. It emphasizes the use of mathematical functions to accomplish tasks. This type of programming usually reduces the reliance on an external state or data. Ultimately, it is easier to debug and more likely to be executed safely alongside other processes.

15-Elixir

Designed by: **José Valim(2012)**

A high concurrency and low latency programming language, it is most commonly used to build scalable and versatile applications, with additional support for low latency and fault tolerance. When your task is to design an app to take loads of requests, Elixir is the answer, for it combines the best features of Erlang, Ruby, and Clojure languages and supports multithreaded applications.

Functional Programming,, Reliability. Scalability

Metaprogramming, Polymorphism, Parallel, Concurrency, Pattern Matching

Used for Web Programming and claim for embedded System

You are able to directly change Abstract syntax tree (AST)

Used by Discord, Pinterest, WhatsApp, Heroku

16- Scala

Scala builds on a foundation laid by Java, making it one of the best programming languages to learn for beginners. It brings a focus on functional programming and concise, flexible syntax. Scala's additions make it easier for expert developers to solve problems more efficiently. Scala is commonly used in data processing, web development, and data engineering.
Twitter first picked up Scala for performance reasons in 2009, and it is perhaps one of the largest Scala users. Most online learning resources for Scala assume you already have at least some experience in programming

17-Clojure

Clojure is a functional programming language based on Lisp, a language that first appeared in 1958 with a unique parenthetical syntax. Clojure added features like immutable data structures to Lisp which makes it easier to write concurrent systems. Companies like Walmart use Clojure for backend services to build systems that can handle large volumes of activity.

Considering Clojure's focus on being useful for concurrency, it's not surprising that it's often used for building services where this is important

18-Groovy

Derived from Java, Groovy was developed by James Strachan and Bob McWhirter. The language improves productivity because of its succinct and easy to learn. Some well-known companies that are using Groovy in their tech stacks are Starbucks, Transferwise, and Craftbase.

19-Haskell

Haskell is a general-purpose programming language named after the American logician and mathematician Haskell Brooks Curry. It is a purely functional programming language meaning it’s primarily mathematical. It’s used across multiple industries particularly those that deal with complicated calculations, records, and number-crunching. Like many other programming languages from this era, it is not overly common to see Haskell in use for well-known applications. With that said, the programming language has been used to write a number of games one of which is Nikki and the Robots.

20-Perl

Perl was created by Larry Wall and is a general-purpose, high-level programming language. It was originally designed as a scripting language designed for text editing but nowadays it’s widely used for many purposes such as CGI, database applications, system administration, network programming, and graphic programming.

21-Objective-C

Developed by Brad Cox and Tom Love, Objective-C is the main programming language used to write software for macOS and iOS, Apple's operating systems.

22-SQL (SEQUEL at the time)-1972

SQL was first developed by IBM researchers Raymond Boyce and Donald Chamberlain. SEQUEL (as it was referred to at the time), is used for viewing and changing information that is stored in databases. Nowadays the language is an acronym – SQL, which stands for Structured Query Language.

Application:

- ❑ Relational Database