è impossibile stabilire quando nasce il primo linguaggio.
Anche solo quando Ada Lovelace studiò gli appunti sull'analytical engine elaborò una specifica precisa che permetteva di calcolare i numeri di Bernoulli, potremmo dire molot forzatamente che quello fosse il primo "programma"
In effetti anche ENIAC e Z1 permettevano una programmazione, configurando a mano collegamenti e switch.
Il primo concetto di loop lo abbiamo per il Mark I, si otteneva direttamente incollando la fine del nastro perforato al suo inizio.
Le basi della programmazione vengono elaborati da Goldistine e Von Neumann (1943-1945) che creano I Flow-Chart.
In ogni caso tutti I dati inseriti nei computer erano codificati in linguaggio macchina, che essi fossere in input o salvati in memoria (EDVAC e codice memorizzato).
L'allontanamento dal linguaggio macchina avviene nel 1949 su una macchina detta EDSAC, con delle istruzioni macchina chiamati Initial Orders, una forma mnemonica del codice operativo delle varie istruzioni (una lettera dell'alfabeto), una forma primitiba di assembler.
Il primo assembler lo troviamo nel BINAC, grazie al suo creatore, Mauchly (BINAC e EDSAC sono versioni successive dedll'eniac), il linguaggio veniva chiamato short code
Il BINAC leggeve proprio espressioni! Nel senso che si scriveva : Y = X + cazzi, cazzi; il programma così scritto veniva poi intepretato dal primo "interprete" e caricato nella memoria del pc.
Ma per molto tempo I computer saranno troppo lenti per essere ancora rallentati da un interprete.

Il primo compilatore.
Corrado Bohm, uno scienziato italiano che ha dato vita al dipartimento di informatica.Definisce (su carta) un linguaggio di alto laivello traducibile con varie funzioni: salti condizionati ed incozionati, uso di subroutine (funzioni). Definisce anche il progetto di un compilatore sciutto nello stesso linguaggio dei programmi che deve compilare (come?).
Ma questo compilatore rimane sempre solo su carta, il vero primo compilatore che verrà inmplementato sarà l'AUTOCODE, sviluppato in inghilterra nel 1952 (due anni dopo Bohm).
Questo compilatore traduceva il codice sulla stessa macchina che poi andava ada eseguire tale codice. (con una perdita di efficienza di solo il 10%).
Still, non ebbe successo. Il problema non era scrivere programmi, ma gestire la poca memoria, trasformare I problemi in codice e debuggare l'hardware.

Assembler, 1954
viene creato un assemblatore, il SOAP (ymbolic Optimal Assembly Program), ed un compilatore (e linguaggio) IT (Internal Translator) che seguiva il classico paradigma a 2 passaggi Assembler intermedio e poi codice macchina.

IT, il primo linguaggio di programmazione
Era di "alto livello" e compilato, soprattutto dimostrò che era possibile creare codice "efficiente" anche usando codice di alto livello. Era anche slegato dall'hardware.

Il FORTRAN, 1957
linguaggio in uso ancora oggi (ultima versione nel 2015)
Fortran si basa su una grossa scommessa: "un compilatore sarà in grado di generare codice efficiente quando il codice scritto da un programmatore?"
Questo era il primo linguaggio ad avere una rigorosa sintassi, slegata dall'assembler e di "alto livello". Un'ora di lezione bastava ad essere in grado di leggere un programma.

Con molta calma (all'inizio era pieno di bug) il fortan diventa affidabile e rapido. Nel giro di un anno metà delle macchine IBM 704 (su cui era sviluppato il compilatore) potevano essere progtammate in FORTRAN.

Il LISP 1958
Alcune parti del compilatore furono scritte in FORTRAN
LISt Processing language, un linguaggio "funzionale", si poneva quindi di rappresentare e manipolare espressioni simboliche (basato sui lavori di kleene era molto adatto per AI e automi).
È il primo linguaggio ad usare la ricorsione, di conseguenza nascono alcune strrutture dati tipo alberi e liste concatenate.
Dettaglio, la metaprogrammazione. Un programma lisp è esso stesso una lista e di conseguenza può essere manipolato da un altro programma lisp.
LISP era super efficiente anche se interpretato, perchè era vicino alla macchina. Alcune sue funzioni potevano direttamente essere implementate in hardware e poi usava la notazione prefissa (+ 3 2), molto più semplice da leggere per l'hw.

COBOL 1959
COmmon Business Oriented Language. Importante perchè servì a sdoganare la programmazione dall'ambito scientifico. (esempio di sintassi: MULTIPLY X BY X GIVING X-SQUARED)
Il cobol venne sviluppato in ambito governativo (USA), tanto che il dipartimento della difesa obbligò le aziende a fornire COBOL nelle loro macchine.
Di conseguenza il COBOL doveva essere molto portabile, cosa facile adesso, ma non al tempo.
Il COBOL divenne il linguaggio più usato al mondo (ma solo in ambito commerciale, in ambito scientifico era più odiato di quanto venga odiato l'html adesso).
Cobol è purtroppo ancora in uso, ma sta sparendo.

ALGOL 60, 1960
ALGOrithmic Language, non è più in uso e nessuno lo conosce ma è stato un linguaggio fondamentale per lo sviluppo dei nuovi linguaggi, con più di 70 dialetti.
Il FORTRAN era più efficiente, ed il COBOL + portabile. spesso si scrivevano algoritmi in algol per poi tradurli.
Caratteristiche del fortran:
I metodi di pasasggio delle procedure (call by value, call by reference e call by name) come call by value, ma più sofisticato. Ed abbandonato
la dichiarazione esplicita del tipo di variabili (integer, boolean,real, double, long, array)
primitive di controllo strutturate (if then else, while, for, do)
nascita di blocchi di istruzioni delimitati da begin- end(metodi) con variabili locali.
Intruduzione di una notazione formale, la BNF (Backus Naur Form). Una notazione per la definizione di grammatiche context free.
E qui si torna ad LFT, perche nel 1961 grazie all'algol iniziano nuovi studi riguardo alle grammatiche. Si conosceva già la forma normale di chomsky ma nasce il concetto di parsificatori (controllo della correttezza sintattica). Prima parser LL e nel 1965 praser LR grazie a Knut.

BASIC 1964
John Kemeny e Thomas Kurtz concepiscono il Beginner's All-purpose Symbolic Instruction Code, disprezzato perchè troppo "semplice", portava I programmatori a scrivere programmi mal strutturati.
Nasceva con l'intenzione di peremettere a chiunque (matematici, scienziati, etc) di scrivere programmi semplici, fu inizialmente pensato solo per la scrittura di algoritmi di calcolo matematico.
Avrà subito successo, ma il vero boom lo farà neglio anni '70 con la nascita prima dei microcomputer e poi del PC, in quanto permette a studenti ed hobbysti di programmare.

BASIC diventa anche un ambiene di lavoro (tipo shell linux), sia interprete di comandi che linguaggio di programmazione (funzionava così l'apple II ad esempio)
Il basic viene modificato ed aggiornato dai produttori di computer, soprattutto da IBM, il culmine del successo di questo trand sarà nel 1991 quando microsoft sviluppera Visual BASIC.

I linguaggi strutturati
ALGOL come BASIC permetteva su uno stile di programmazione non strutturata (Esistevano ed erano molto usati I GOTO).
Tutti i linguaggi del tempo incoraggiavano uno stile di programmazione molto "spaghetti".
Nel 1966 gli italiani Corrado Böhm e Giuseppe Jacopini dimostrano un teorema (teorema Bohm-Jacopini) che asserisce che qualsiasi funzione computabile può essere computata tramite il solo uso di sequenze di istruzioni eseguite una dopo l'altra, istruzioni di selezione (del tipo if then else) oppure istruzioni di iterazione (del tipo while do). I goto non sono necessari.
Si scatena una guerra fra chi è contro il goto (Dijkstra) e chi è a favore (Knut, Kernighan o Ritchie).
Da questa guerra uscirà vincitore l'anti-goto. per quanto non sia completamente abbandonato nascono molti linguaggi che non ne fanno uso.

Il Simula
Nasce il nonno dei linguaggi di programmazione OO. Il Simula I (1995) è quasi un dialetto ALGOL.
Nel 1966 i creatori Ole-Johan Dahl e Kristeen Nygaard iniziano a sviluppare il Simula 67; lo basano sul concetto di Classe e sottoclassi (proprio dell'ALGOL) e lo estendono definendo una gerarchia.
I paradigmi OO erano tutti presenti: Oggetti, new, Classi, sottoclassi, ereditarietà.
Però per implementare il concetto di oggetto è naturalmente necessario implementare il concetto di puntatore (ref in Simula) e la garbage collection.
Simula implementa molto bene anche il parallelismo, è il primo linguaggio a farlo

Il Pascal 1971
Prodotto da Niklaus Wirth in modo da "migliorare" l'algol, ne viene infatti considerato il successore.
Il Pascal risolveva il vecchiume di algol (non c'erano ne Char ne puntatori ed i compilatori ALGOL erano difficili da scrivere e legatissimi all'hardware).
Una cosa fondamentale del Pascal fu un ricco sistema di tipi, controllato da un rigido type checking in fase di compilazione.
Riguardo al compilatore! Questo non compilava in codice macchina bensì in un linguaggio intermedio che poi veniva interpretato (stile java) da un virtual machine.
Notevole il dialetto turbo Pascal. Un linguaggio poco diverso da Pascal che compilava molto più in fretta, da qui il nome.
All'apice corrisponde anche il declino. Per quando sia Apple che poi Microsoft usino il Pascal come linguaggio standar dei loro sistemi la diffusione di Unix sta facendo si che il C sia sempre più popolare

Il Prolog 1972
Programmazion Logic (fancese)
Come il lisp rimane confinato ai centri di ricerca, lo ricordiamo perché è il predecessore dell'ultimo paradigma della programmazione: la programmazione logica.
Il prolog è un dimostratore automatico di teoremi (wut, really?) e grazie a questa sua possibilità sarà la base su cui partirà la "5th generation"
Nel 1981 il governo giapponese inizia un progetto per la costruzione di supercomputer basati sul parallelismo e le AI. L'assemly di questa nuova generazione doveva essere il prolog.
Vennero quindi sviluppate CPU che fossero ottimizzate per svolgere "inferenze" logiche e che sfruttassero al massimo il parallelismo.

Ma questo progetto fu un flop completo, per due motivi:
1 negli anni ottanta nacquero i RISC, ed i processori CISC vennero sorpassatti, anche quelli di 5 generazione.
2 la logica non è per nulla facile da implementare su hw, è quasi impossibile da ottimizzare

Lo Smalltalk ‘70
La vera programmazione ad oggetti. Sviluppato a Xerox PARC (Palo Alto Research Center)
Nasce per la necessita di sviluppare un computer progettato da Alan Kay, il Dynabook.
Il Dynabook avrebbe dovuto essere l’equivalente dei tablet attuali, un posto dove tenere numeri, dati personali, libri, semplici giochi. Tutto cio con una dimensione massima di una borsa.
Smalltalk avrebbe dovuto essere sia interprete dei comandi che linguaggio con cui erano scritti i programmi.
In smalltalk tutto è oggetti, anche le classi sono oggetti cazzo. Porta all’estremo il discorso di incapsulamento. L’unico modo di accedere ai dati degli oggetti è attraverso metodi (getter e setter)
Con lo Smalltalk nasce anche l’idea di un IDE, un editor sviluppato specificamente per quel linguaggio.

ADA 1974 (entra in uso nel 1980), un linguaggio per dominarli tutti
Sviluppato di nuovo dal dipartimento della difesa USA per unificare un po’ la miriade di linguaggi e dialetti, dato che ogni gruppo di ricerca e sviluppo usava un linguaggio diverso.
ADA parte male, con compilatori lenti e che producono codice inefficiente.
Dal 1991 al 1997 il DoD impone l’ada in tutti i suoi progetti.
ADA prende le caratteristiche migliori di ogni linguaggio: sintassi e struttura ALGOL, i tipi del Pascal, concorrenza e parallelismo del Simula.
Molta attenzione era posta al produrre il minor numero di errori possibile.
Eppure ADA non diventa così popolare, per quanto sia diffuso ancora oggi è circa al 30° posto in ordine di utilizzo

Il C anni ‘70 (<3)
Il C è il linguaggio più longevo e più di successo nella storia dell’ informatica, con innumerevoli discendenti.
Il suo vero punto di forza è la portabilità e l’efficienza, quindi è molto adatto per applicazioni embedded o compilatori ed interpreti (si veda Cyton).
Ma a anche la sua modularità da una gran spinta al C.
Altre caratteristiche positive del C sono: la compattezza, i tipi flessibili (più del pascal ma comunque presenti), gestione della memoria (grazie ai puntatori) e possibilità di agire sui singoli bit.
Insomma il C unisce la granularità e la potenza di assembly con la flessibilità dei linguaggi di alto livello. Di contro è molto facile fare errori in C.
Viene creato da una collaborazione fre MIT, Bell Labs e General Electrics mentre si cerca di creare un sistema operativo.
Dennis Ritchie, Ken Thompson e Brian Kernighan lavoravano su MULTICS; un progetto che perde i fondi a metà sviluppo. Senza fondi continuano il progetto nel loro tempo libero cambiando il nome in Unix per scherno.
Per lo Unix viene sviluppato un linguaggio di alto livello, il B (una semplificazione di un dialetto ALGOL chiamato BCPL).
È proprio dal B che deriva l’uso del ; e delle parentesi graffe per delimitare parti di codice.
Il B non era tipizzato, nel 1972 Ritchie aggiunge i tipi e scrive un compilatore, per il C.
Unix viene diffuso gratuitamente, insieme ai suoi sorgenti, scritti in C.
Negli anni ‘80 Microsoft addotta ufficialmente il C come linguaggio di riferimento (e poi il C++).
Riguardo al C++, nasce nel 1984 con il nome “C with Classes”.

C++ mantiene tutte le qualità del C (con qualche differenza sintattica) ed è compatibile. Purtroppo questo è sia un vanto che un ostacolo, in quanto non usando nativamente classi è un po' macchinoso.
Objective-C e C# sono le implementazioni di C di rispettivamente Apple e Microsoft.

Java 1990, lanciato 5 anni dopo
James Gosling, nell'azienda Sun Microsystem, si mette a progettare una Smart TV (prima di internet, sulla TV via cavo); serviva quindi un linguaggio in cui sviluppare applicazioni che dovevano essere scaricate dalla rete e soprattutto dovevano girare su sistemi con poca CPU (il C++ consumava troppe risorse).
Il nome inizia come C++ ++ –, poi passa ad Oak, poi java (sai già perché).
Nel 1993 la nascita del WWW manda all'aria i piani della Sun, ma loro si sanno adattare riconvertendo Java dalle "Top box" ai browser web.
Sui browser poteva essere integrata una JVM e tramite questa possono essere scaricate e lanciate delle "applet" (Write Once Run Everywhere).
Dal web il Java si espanse grazie alla sintassi simile a C e C++, alla sua portabilità (bytecode) ed alla sua semplicità di scrittura.

Lo Scripting '90
Script: programmi corti eseguiti da un interprete (esempio più ovvio, Unix shell).
Uno script è sempre legato ad un ambiente software, quinidi è sempre relativo al suo interprete; ogni linguaggio può diventare un linguaggio di scripting, ma ti ci voglio vedere a cambiare directory in java, alcuni linguaggi sono + orientati allo scritpting.
Negli ultimi anni si sono molto diffusi i linguaggi di scripting general purpose, linguaggi che uniscono le caratteristiche strutturali dei classici linguaggi (struttura, object-oriented, etc) alle caratteristiche dei linguaggi di scripting (prototipazione veloce, conttrollo di tipi a runtime ed interprete)
Perl, il più vecchio, 1987, nasceva per fornire un nuovo linguaggio per manipolare testi e file in ambiente Unix, sintassi simile al C
Python (attualmente il linguaggio più usato), nasce nel 1991, da Guido von Rossum. È object oriented ma usa una sintassi particolare (li conosci l'indentazione). La sua implementazione più diffusaè scritta in C (CPython)
PHP (PHP Hypertext Preprocessor), nasce nel 1994 da Rasmus Ledorf. È il linguaggio to -go per la programmazione server. È peartito come linguaggi di scripting ma si è evoluto tantissimo.
JS sviluppato nel 1995 in 10 giorni (e cazzo se si nota). Nasce come alternativa a java per scrivere apllet senza dover usare la JVM.
Qui nasce una interessante storia poiché nella guerra tra browser (Netscape vs Explorer) windows crea la sua versione, Jscript. Ed è per questo che scrivere codice JS compatibile con IE è molto scomodo.
Ruby 1995
sviluppato da Yukihiro Mastumoto, per creare un linguaggio semplice ma OO. Molto diffuso nelle applicazioni web.