Caller & Callee have same status
→ no concept of master & slave
→ coroutine

in most imperative models

Has classes → abstractions (OOP)

ALGOL 68 → orthogonality → simple data types
→ User defined data types [Variable arrays]
→ was carried over to others

→ descendants: PASCAL (teaching Programming language

→ C (from BCPL, B & ALGOL 68)
[1972] → introduced as a system language

1989 - ANSI C
1999 - ISO Standarized

1980 - C compiler shipped with UNIX

Prolog: declarative, rule based,
based on predicate calculus
AI - application, Intelligent DB

Fact statements

Mother(x, y)
Father(m, n)

Rule statements

grandparent(a, b) = parent(a, c), parent(c, b)

Prolog DBs can be interactively queried with good statements
eg: Father(m, n) T or F?

Ada - 1975 (DoD) Dept of Defence — Exception Handling

Possible to write generic methods
↳ not specific to a type

1995 - OOP concepts

Smalltalk - First fully object oriented PL

* Everything is an object * (Java has primitives too, but not Smalltalk)

⇒ You need an object to use methods

⇒ First PL which supported GUI dev
(Simula 67 did, but not OOP i guess)

C++: dynamic binding
templating
both procedural & ~~funct~~ OOP
Multiple inheritance
(1983 think)

Java doesn't support

Java - 1990 (based on C/C++)
Obj C - 2009
Delphi
Go

→ JIT (Certain parts of bytecode can be converted into machine code.)
→ ~~GC~~
→ ~~Lot~~ garbage collection, array check

Ruby: Dynamically add methods to class (?)

(C#) → (VBS & C++)
for GUI

.NET → interpreted, JIT compiled
PLs under .NET have a library

Markup languages

HTML: Hypertext Markup language.
Describes Display properties

XML = like HTML, uses HTTP

But Describes data present, instead of Display

XSLT → Stylesheet → Language translations

(Extensible)

JSP - Java Server Pages.

↳ gets the code from server → executed → result

using HTML specifications,

Displays as a webpage