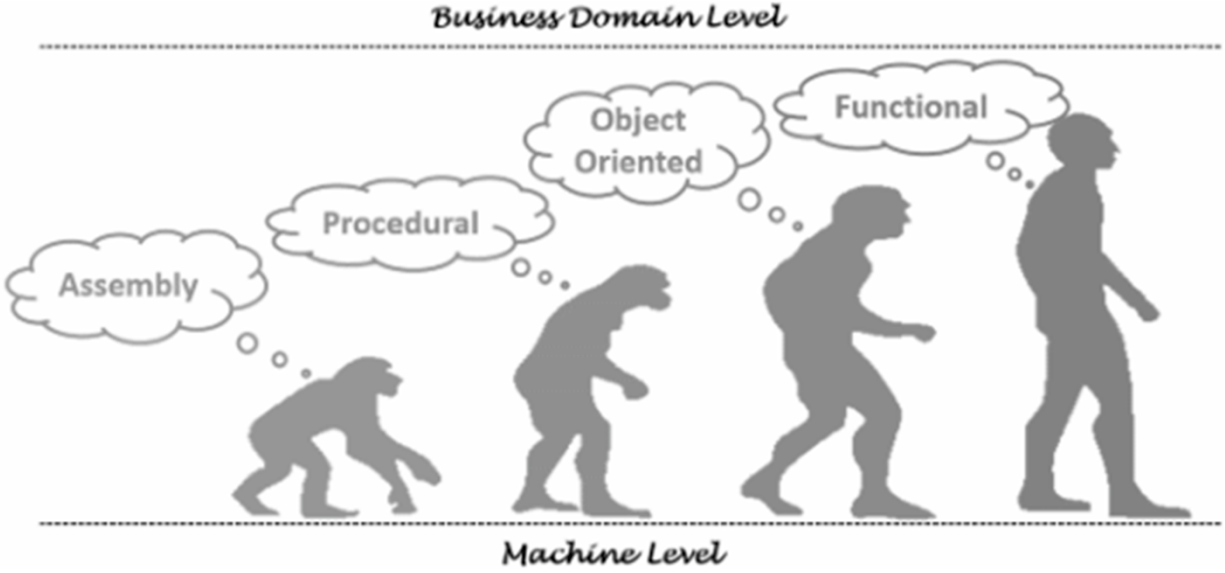
M. Caramihai, © 2022

PROGRAMAREA ORIENTATA OBIECT

CURS 1

A gandi "orientat obiect"



Evolutia programarii

- Cod masina
- Limbaj asamblare
- Limbaje independente de masina
- Proceduri si functii
- Obiecte
- Servicii

Istoric

- ~1960
  - SIMULA 1 (1962) si Simula 67 (1967), Norwegian Computing Center in Oslo, Norway
  - Smaltalk (1970) – XEROX
  - "C with Classes", predecesorul C++, 1979, Bjarne Stroustrup
  - 1995, Java dezvoltat de James Gosling la Sun Microsystems.
- 1980 – Revista *Byte*
- Prima conferinta dedicata OOP, Portland, Oregon, USA

Generalitati

- POO a evoluat din *programarea structurata (PS)*
- POO este:
  - O idee revolutionara
  - O evolutie normala
- Aplicabilitate
  - POO: sisteme distribuite
  - PS: fluxuri de date, descompunere de procese

Limbaj si gandire (1)

- M. Sapir (1956): “Oamenii se gasesc la cheremul unui limbaj particular care reprezinta mediul de comunicare in cadrul unei societati”
  - ⇨ Limbajul influenteaza gandirea
- d.e. vocabularul extensiv (eschimosii si zapada)

- **Concluzie:** un limbaj OO poate simplifica dezvoltarea unei solutii OO

Limbaj si gandire (2)

- Exemplu: analiza secventei DNA: fie un vector cu N variabile in cadrul careia trebuie determinata o secventa repetitiva de lungime M
- Rezolvare intr'un limbaj standard:

  M x N x N operatii
- Rezolvare intr'un limbaj OO

  M x N x log(N) operatii
- *Observatie*:

  Limbajul OO NU este mai rapid decat limbajul structurat

Ipoteza lui Church & postulatul lui Whorf

- **Ipoteza lui Church / Church-Turing (1936):** orice rationament pentru care exista o “procedura efectiva” (i.e. are la baza un algoritm) poate fi realizat de o masina *Turing*.

Reguli:

- Numarul de instructiuni trebuie să fie finit.
- Iesirea ar trebui să fie obtinuta dupa parcurgerea unui numar finit de pasi.
- Instructiunile pot fi implementate în lumea reala.
- Instructiunile nu ar trebui sa necesite o intelegere complexa.

https://www.youtube.com/watch?v=0D7yInuKvKs

Ipoteza lui Church & postulatul lui Whorf

- **Postulatul lui Whorf / Sapir-Whorf (1956):** este posibil ca un individ, ce comunica intr'un limbaj, sa nu poata comunica ganduri / idei intr'un alt limbaj

Astfel, se postuleaza faptul ca natura unei anumite limbi influenteaza gandirea obisnuita a vorbitorilor ei. Diferitele modele de limbaj genereaza diferite modele de gandire. Aceasta idee contesta posibilitatea reprezentarii perfecte a lumii cu limbajul, deoarece recunoaste ca mecanismele oricarei limbi conditioneaza gandurile comunitatii de vorbitori ai acesteia.

https://study.com/academy/lesson/sapir-whorf-hypothesis-examples-and-definition.html

Consecinte

- Pentru **Ipoteza lui Church**
  - Orice limbaj in cadrul caruia poate fi simulata o masina *Turing*, este suficient de performant pentru a permite implementarea oricarui algoritm
  - Toate limbajele de programare sunt identice

- Pentru **Postulatul lui Whorf**:
  - Pot exista idei ce pot sa fie exprimate mai bine intr'un limbaj decat intr'altul

- **Consecinta**
  - George Steiner (1975 - *After Babel*): orice act de comunicare umana poate fi vazut ca implica o traducere

O noua paradigma

- Programul OO creaza obiecte ce incapsuleaza date si procedure ce vor actiona asupra acelor date
- Un POO este structurat ca o comunitate de agenti / obiecte ce interactioneaza intre ele
- Fiecare obiect produce un serviciu sau realizeaza actiuni
  - Exemplul florarului

Caracteristicile POO

- O noua metoda de proiectare a aplicatiilor pornind de la o reprezentare naturala
- Programele sunt gandite a avea *componente* ce functioneaza independent, fiecare cu propriile proprietati si functionalitati
- Un sistem este proiectat ca o interconexiune de asemenea *componente*.

Caracteristicile programarii structurate

- Programele sunt gandite strict secvential
- Se utilizeaza *branching conditions* in cazul implementarii deciziilor
- Se utilizeaza functii / proceduri pentru implementarea secventelor *repetabile*.
- **Sinteza:** programele definesc date, dupa care apeleaza subprograme pentru a prelucra acele date

Limitarile programarii structurate

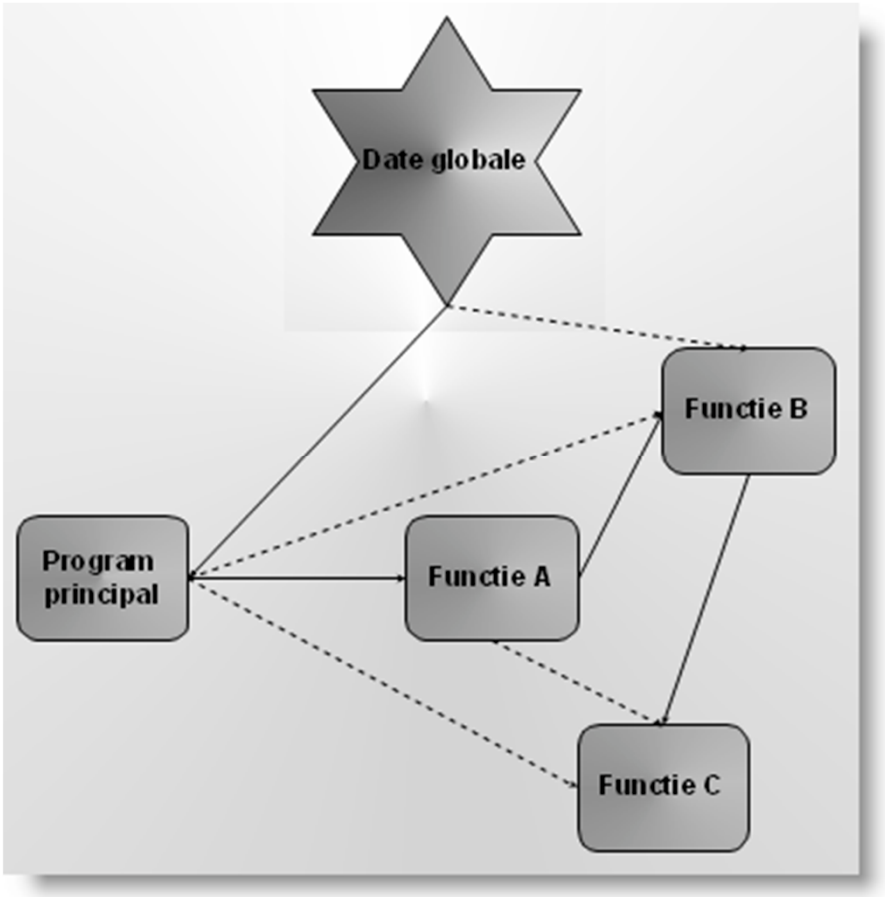
- In viata reala evenimentele nu sunt intotdeauna secventiale
- Nu este simplu de gasit solutii secventiale pentru problemele reale
- Codul aplicatiei nu poate fi re-utilizat si depanat independent
- Codul nu se poate “adapta” la situatiile in schimbare

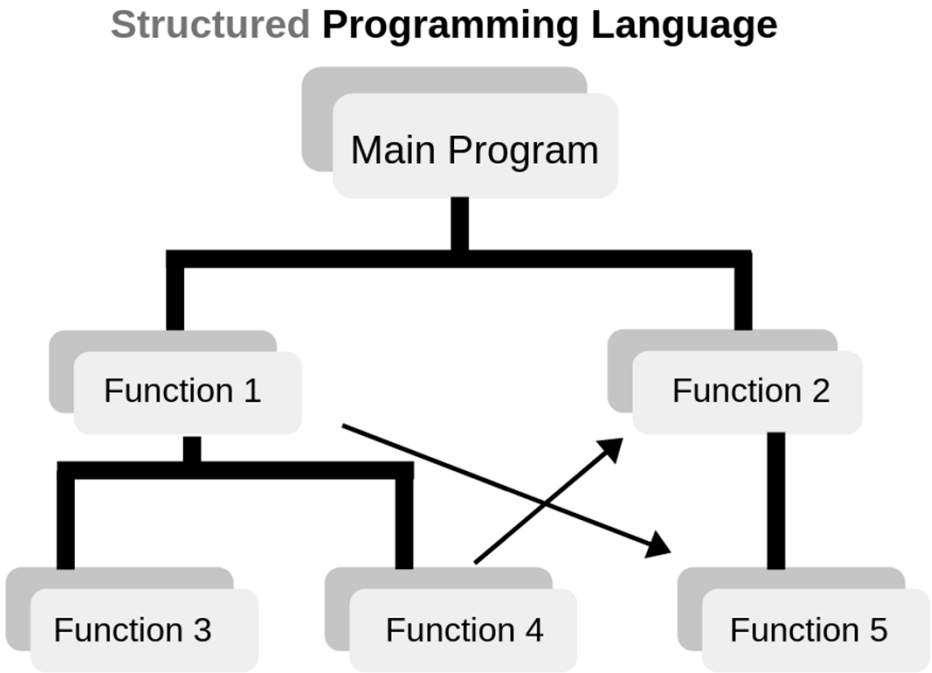
- → Abordari OO sunt benefice in cadrul aplicatiilor de dimensiuni mari (>1000 linii de cod)

POO – un nou model de gandire

- Actiunile sunt initiate in POO prin transmitere de mesaje de catre un agent (obiect) responsabil pentru o actiune
- Programele sunt gandite ca interactiuni / asocieri de obiecte
- In cadrul POO nu trebuie “anticipat” orice – ci trebuie doar utilizate serviciile obiectelor

Programarea structurata





POO vs. Programarea structurata

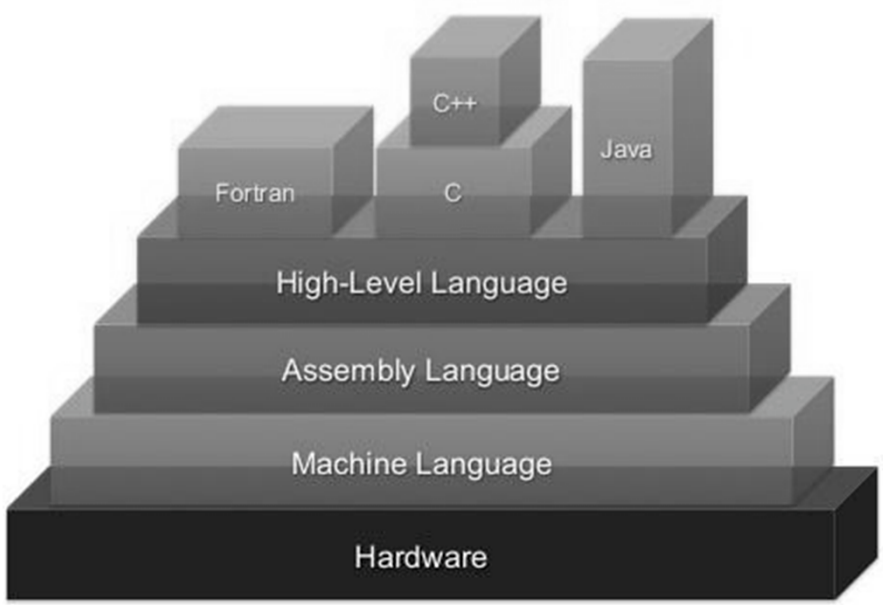
STRUCTURED PROGRAMMING VERSUS OBJECT ORIENTED PROGRAMMING

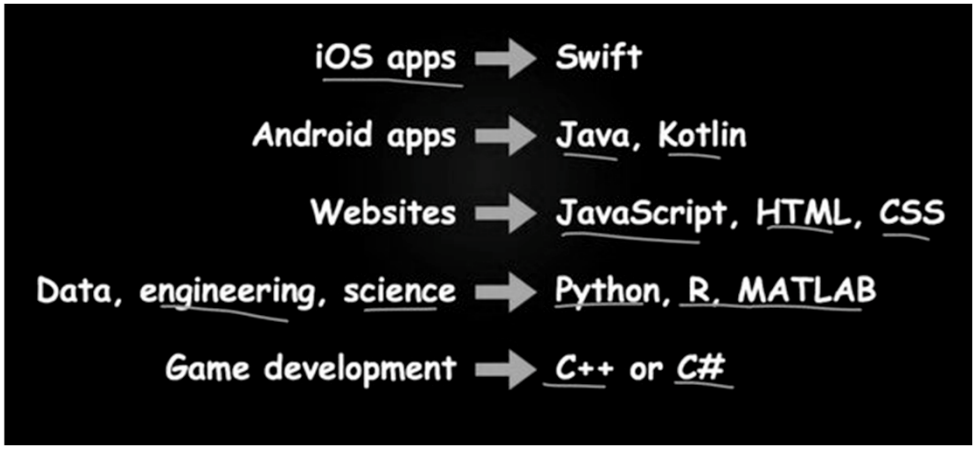
STRUCTURED PROGRAMMING	OBJECT ORIENTED PROGRAMMING
A programming paradigm that divides the code into modules or function	A programming paradigm based on the concept of objects, which contain data in the form of fields known as attributes, and code in the form of procedures known as methods
Focuses on dividing the program into a set of functions in which each function works as a subprogram	Focuses on representing a program using a set of objects which encapsulates data and object
Difficult to modify structured programs	Easier to modify Object Oriented programs
Main method communicates with functions by calling those functions in the main program	Objects communicate with each other by passing messages
There are no access specifiers	There are access specifiers such as private, public and protected
Data is not secured	Data is secure
Difficult to reuse code	Easy to reuse code

Evolutia limbajelor de programare (1)

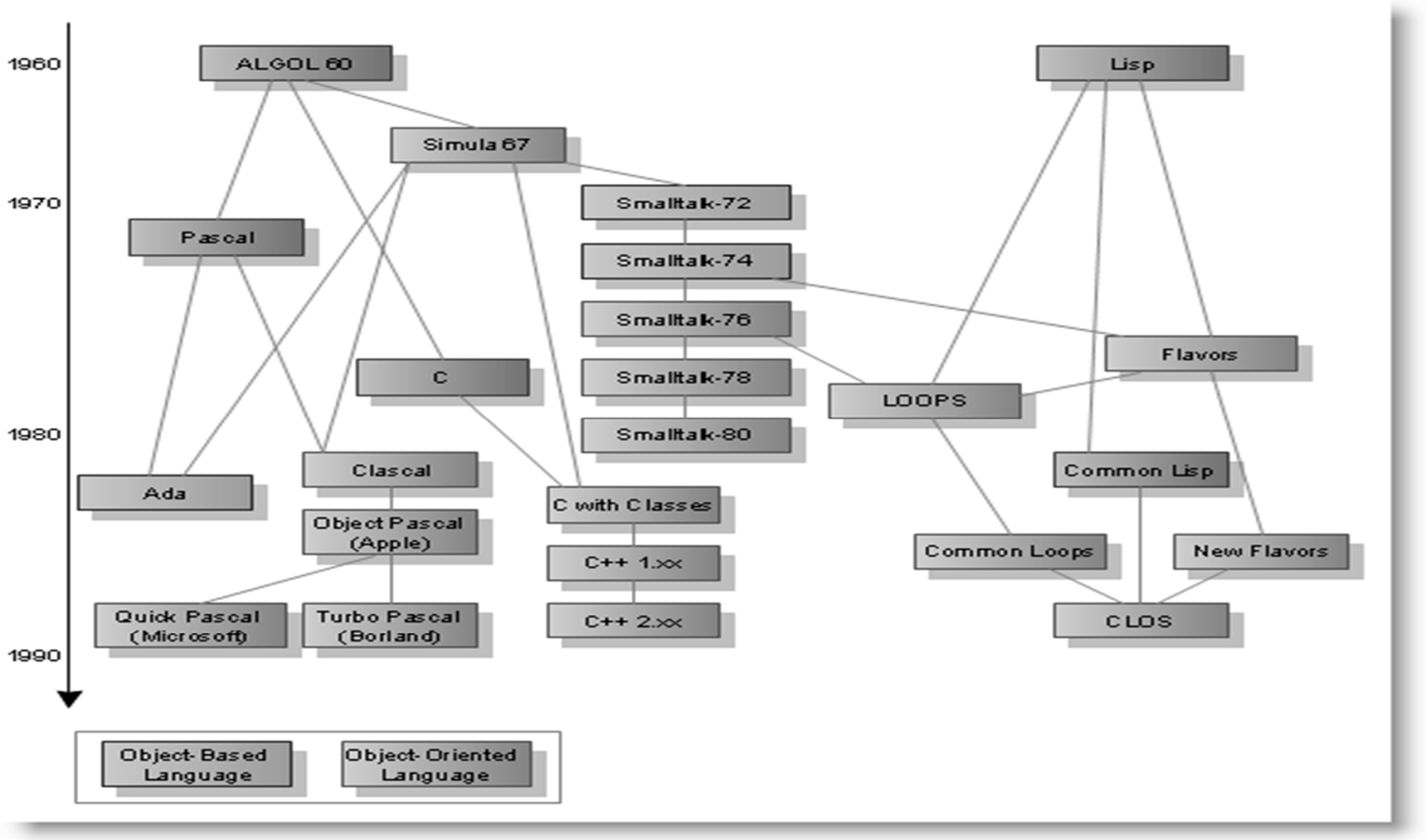
- Generatia I (1954 - 1958) – formule matematice
  - Fortran I, Algol 58, IPL
- Generatia II (1959 -1961) – algoritmi
  - Fortran II, Algol, Cobol
- Generatia III (1962 - 1970) – abstractizare date
  - PL/I, Pascal, Algol 68, C, Fortran77
- Limbaje OO
  - Smalltalk, C++, Eiffel
- Limbaje actuale: Java, Python, C#

Evolutia limbajelor de programare (2)



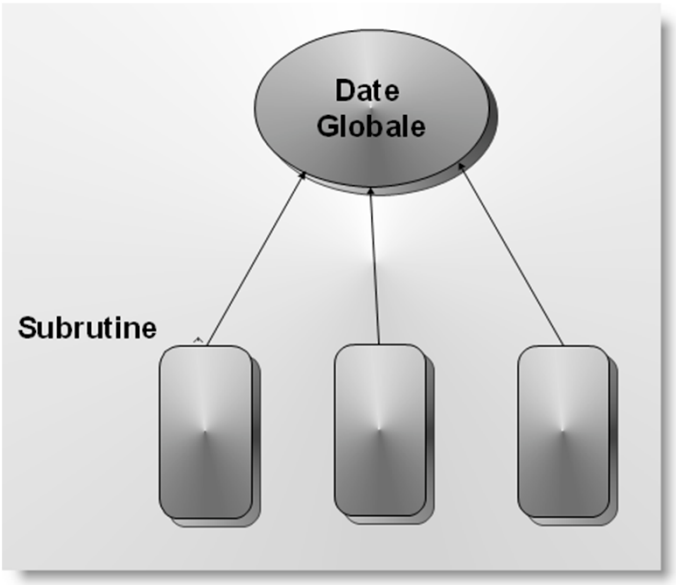


Evolutia limbajelor de programare (3)

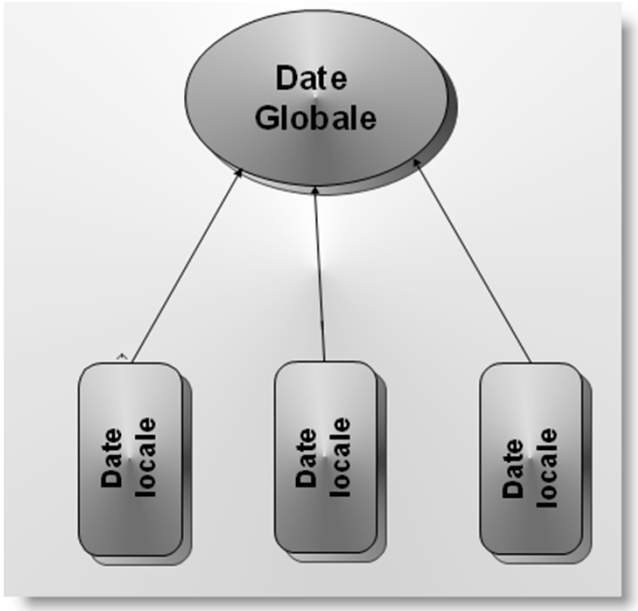


Evolutia limbajelor de programare (4)

Generatia I



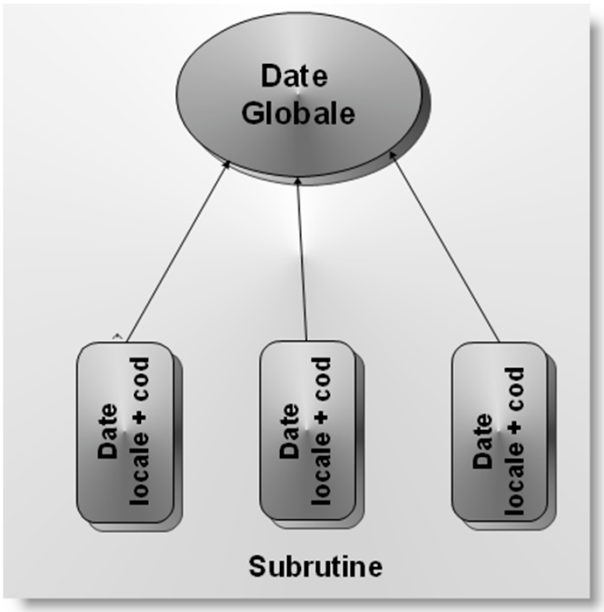
Generatia II



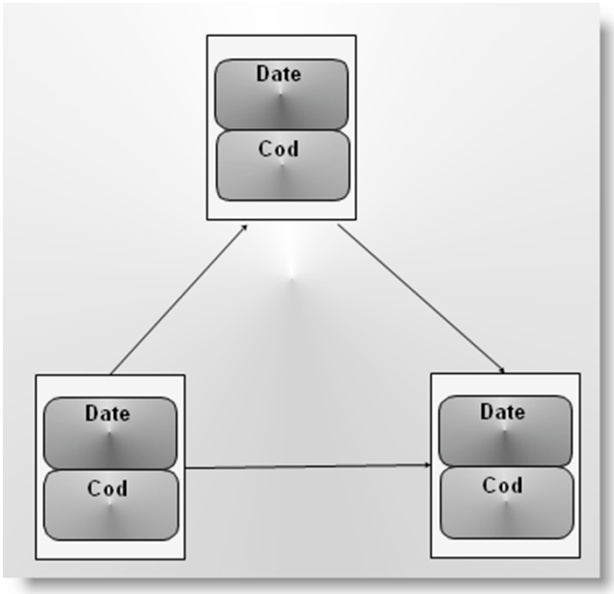
- Mecanisme de “transfer parametrii”
- Dezvoltare stracturi de control
- Subprograme

Evolutia limbajelor de programare (5)

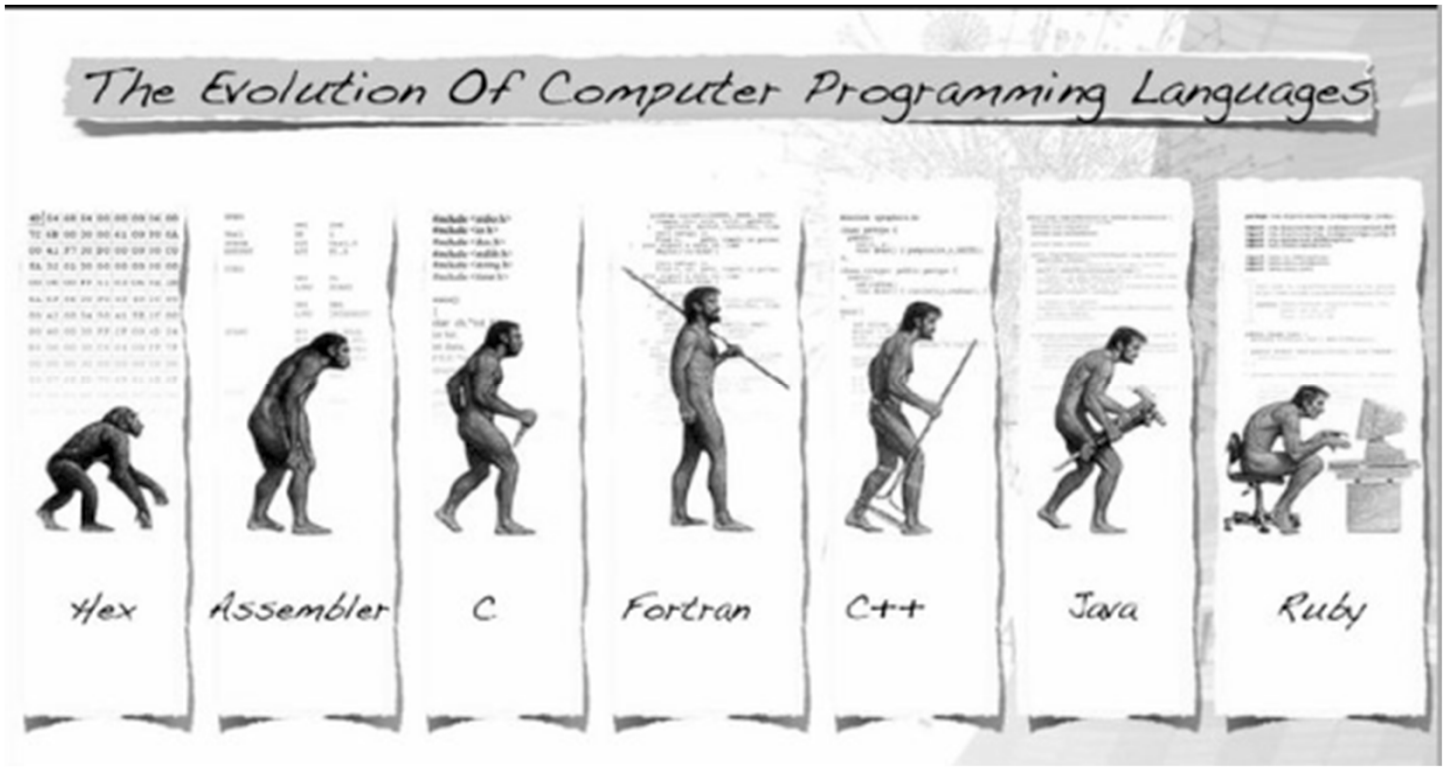
Generatia III



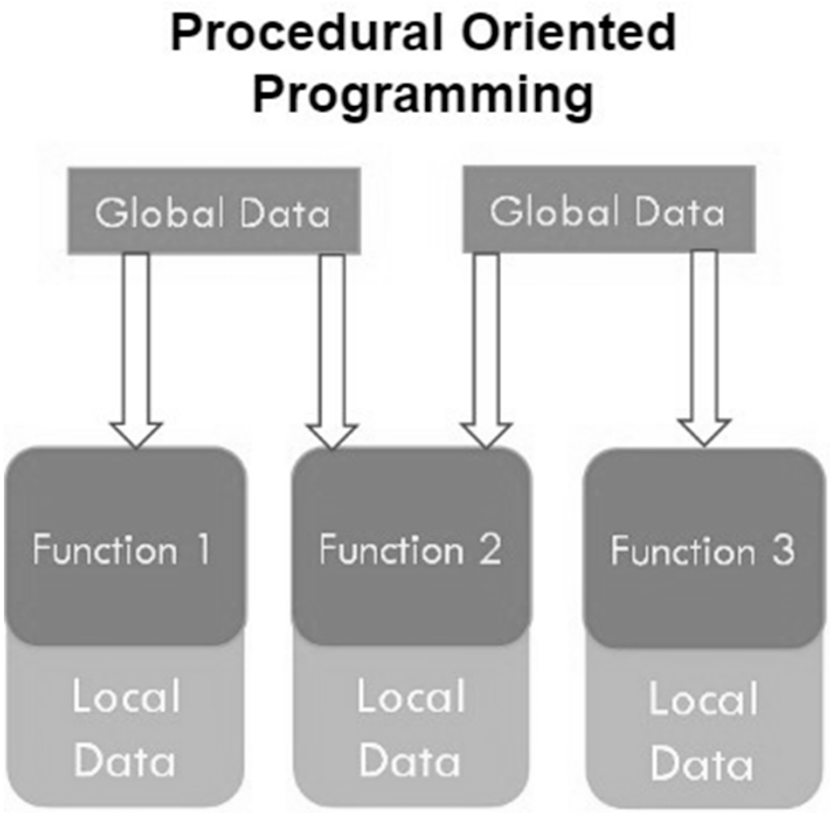
Generatia IV

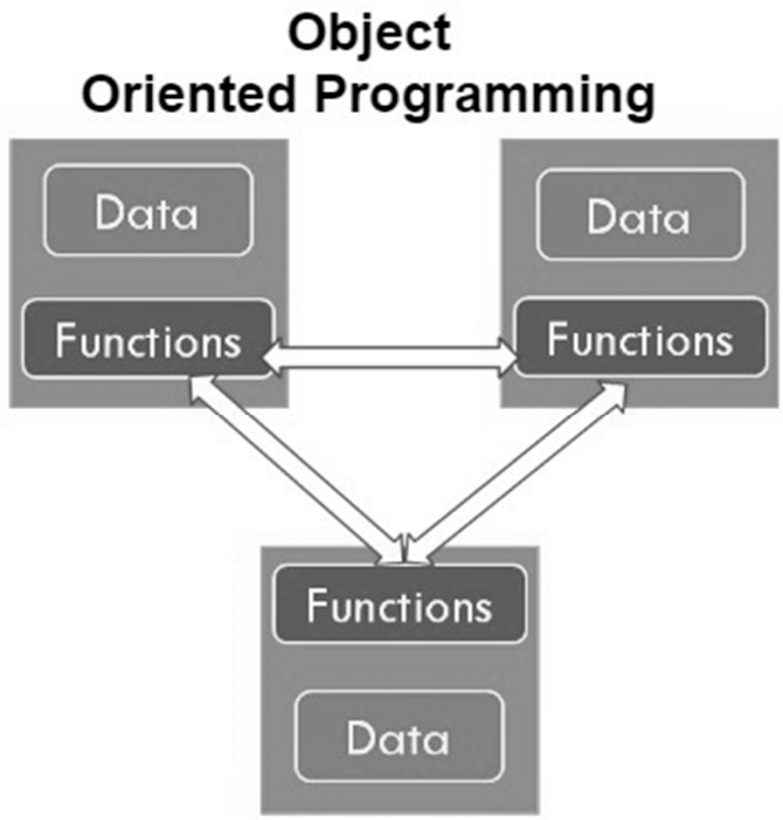


Evolutia limbajelor de programare (6)

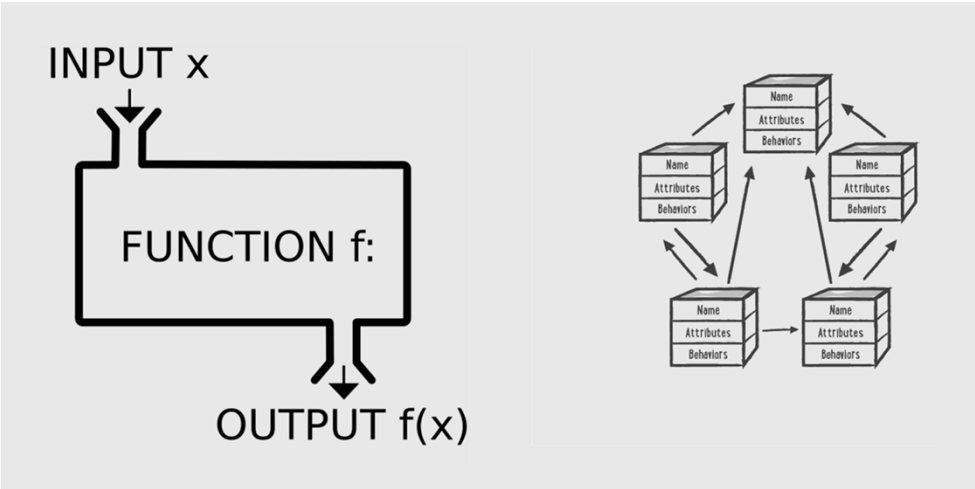


Programarea functionala vs. Programarea procedurala vs. Programarea OO (1)





Programarea functionala vs. Programarea procedurala vs. Programarea OO (2)



FP	OOP
is based on data and their transformations of the functions	is based on objects and models
utilizes immutable data	utilizes mutable data
follows the declarative programming paradigm	follows the imperative programming paradigm
uses recursions for iterations	uses cycles for iterations
allows parallel programming	allows parallel programming without high-level abstractions
if/else statements and *switch/case* operators as well as even more powerful pattern matching	*if/else* statements and *switch operators*
use of commands randomly	use of commands in a set order

Avantajele POO

- Analiza si proiectare mai facila
- Reutilizare cod
- O mai buna mentenanta a programelor
- Reducerea codului recurrent
- Mascarea informatiilor
- Robustete sporita
- O mai buna securitate a aplicatiilor

Concluzii … istorice

- **“I think there is a world market for maybe five computers.”**
  - Thomas Watson, chairman of IBM, 1943
- **“Computers in the future may weigh no more than 1.5 tons.”**
  - Popular Mechanics, 1949
- **“There is no reason anyone would want a computer in their home.”**
  - Ken Olson, president, chairman and founder of Digital Equipment Corp., 1977