


Universitatea "Politehnica" din București  
Facultatea de Electronică, Telecomunicații și  
Tehnologia Informației



# Programarea Calculatoarelor (limbajul C)

## Curs 1 – Introducere

Prof. Bogdan IONESCU

2016-2017

## Bibliografie

[1] Curs,

[2] C. Dan, D. Burileanu, "**Introducere în programarea calculatoarelor. Limbajul C**", Editura Printech, București, 2001.

[3] D. Burileanu, C. Dan, M. Pădure, "**Programare în C. Culegere de probleme**", Editura Printech, București, 2004.

[4] Îndrumarul de laborator (disponibil la laborator),

[5] Orice altă carte de C/C++, Internet !

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU, 2016-2017

## Cuprins

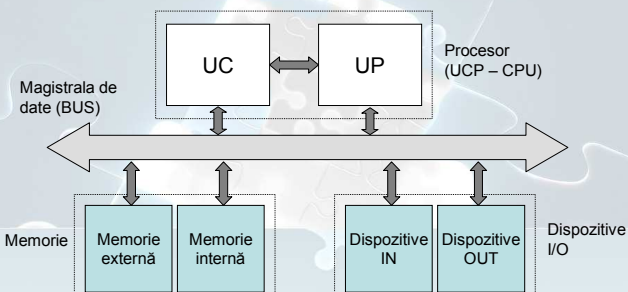
- 1.1. Sisteme de calcul (modul de funcționare)
- 1.2. Hardware și Software
- 1.3. Limbaje de programare (generalități)

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU, 2016-2017 1/35

## 1.1. Sisteme de calcul (modul de funcționare)

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU, 2016-2017 2/35

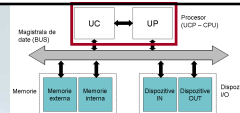
**Sistem de calcul = echipament electronic destinat prelucrărilor complexe ale informației.**



Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU, 2016-2017 3/35

### Procesorul: UC + UP

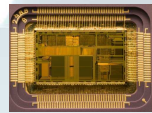
CPU – Central Processing Unit



- **UP** = unitatea de prelucrare, efectuează **operații simple** (aritmetice și logice) asupra unor operanți (date) preluate din memorie; rezultatele → în memorie.
- **UC** = unitatea de control, coordonează funcționarea celorlalte blocuri pe baza unor **comenzi** (instrucțiuni din mem.)

→ parte integrantă a informației transmise calculatorului de către utilizator.

**microprocesor** = o singură capsulă de circuit integrat.



Ex. Intel 80486

Curs Programarea Calculatoarelor, Prof. Bogdan IONESCU, 2016-2017 4/35

## Memoria: internă + externă.

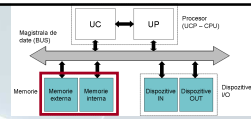
- **Memoria internă** = o colecție de celule de memorare organizată ca o secvență de cuvinte (word) binare (numere).

Un bit = 0 sau 1, un cuvânt = 8 biți, ex.: 00000001, 10101010

> Memoria este organizată matriceal, fiecare locație este identificată cu un număr de ordine numit și **adresă**.

> Din punct de vedere funcțional (nu fizic) memoria este de două tipuri:

- **memorie de date**: conține operanzi, rezultate și rezultate parțiale,
- **memorie de program**: conține instrucțiunile (comenzile) care asigură prelucrarea corespunzătoare a datelor.



## Memoria.

> Organizarea datelor:

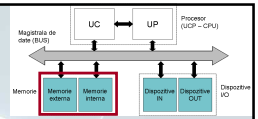
în general cuvinte pe 8 biți = **octet** sau **byte**.

> Prefixele folosite pentru a desemna capacitatea memoriei:

1 **byte** = 8 biți  
1 **kilo-byte** (kB) = 1024 bytes =  $2^{10}$  bytes  
1 **mega-byte** (MB) = 1024 kB =  $2^{20}$  bytes  
1 **giga-byte** (GB) = 1024 MB =  $2^{30}$  bytes  
1 **terra-byte** (TB) = 1024 GB =  $2^{40}$  bytes etc.

> Atenție: 1Mb  $\neq$  1MB. De regulă capacitatea de transfer a rețelelor este exprimată în Mb (mega-biți):

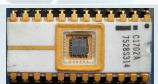
1Mb =  $2^{20}$  biți, 1MB =  $2^{20}$  bytes =  $2^{20} \times 8$  biți



## Memoria.

> Din punct de vedere fizic, memoria internă este de două tipuri:

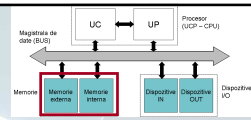
- **ROM** – Read Only Memory: este o memorie permanentă. Conținutul acesteia rămâne intact chiar în lipsa alimentării cu curent. Nu poate fi scrisă de programe.
- **RAM** – Random Access Memory: este o memorie volatilă al cărei conținut se pierde în momentul în care se oprește alimentarea acesteia cu curent. Este o memorie de lucru.



Ex. EPROM Intel 1702



Ex. Kingmax DDR 512MB

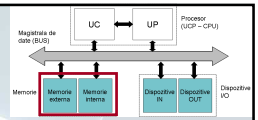


## Memoria.

- **Memoria externă** = un suport extern de stocare a informației. De regulă este folosită pentru a păstra cantități mari de informație.

Exemple:

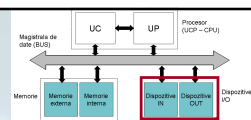
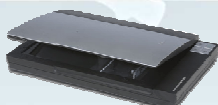
- **FD** (floppy disk drive): 3½-inch, 1.44MB
- **CD-ROM** (compact disk - read only memory): <700MB
- **HD** (hard disk drive): 500GB, etc.
- **FD** (flash drive): 256GB, etc.



## Dispozitivele I/O (In/Out).

- **Dispozitivele de intrare** = dispozitive electronice ce permit introducerea informației într-un sistem de calcul (date și/sau programe). Permit interacția om-mașină.

Exemple: tastatura, mouse-ul, microfonul (sunet), camera WEB (sunet-imagine), tableta grafică (mișcări ale mâinii), scanner-ul (documente fizice), scanner de amprentă (fingerprint), etc.

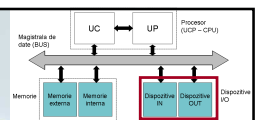


## Dispozitivele I/O (In/Out).

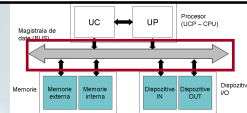
- **Dispozitivele de ieșire** = dispozitive ce permit transmiterea către operator a informației (rezultate, date, etc.).

Exemple: monitorul (informație vizuală), imprimanta (informație pe suport fizic), placa de sunet (informație audio), etc.

> Există bineînțeles și **dispozitive mixte**, In/Out, de exemplu: placa de rețea (primește și transmite informații), Fax Modemul, memoria externă poate fi văzută ca un astfel de dispozitiv.



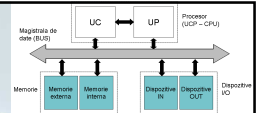
## Magistrala de date (BUS).



> Circuitele de legătură dintre unitatea centrală de prelucrare, memoria internă și dispozitivele periferice formează **magistrala de date (BUS)**.

> BUS-ul este astfel un sub-sistem ce transferă informația în interiorul sistemului de calcul, între componentele acestuia.

**Unitatea de prelucrare (UP):**  
conține circuite logico-aritmetice și registre de memorie.



> **Registreele** sunt tot locații de memorie ce sunt interconectate "strâns" cu circuitele logico-aritmetice (de regulă pe același integrat pentru a fi accesate rapid).

> Operațiile ce pot fi efectuate sunt:

- operații aritmetice: +, -, x, /,
- operații logice: conjuncție (ȘI), disjuncție (SAU), negație:

$$(13 > 10) \text{ ȘI } (5 < 3) = 0 \text{ (Fals)}$$

$$(13 > 10) \text{ SAU } (5 < 3) = 1 \text{ (Adevărat)}$$

$$\text{NOT } (10 > 5) = 0 \text{ (Fals)}$$

Operațiile ce pot fi efectuate (continuare):

- comparații: =, ≠, <, >, ≤, ≥,

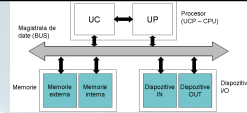
- deplasări ale biților de pe o poziție pe alta în cadrul aceluiași cuvânt binar:

$$9 \text{ (baza 10)} = 2^3 + 2^0 \rightarrow 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \text{ (binar)}$$

$$128 = ??? \text{ dar } 255 = ???$$

$$\text{shift left } (0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1) = 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0$$

$$\text{shift right } (0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1) = 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$$



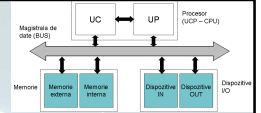
**Unitatea de control (UC):** corelează operațiile elementare ale fiecărui bloc în cadrul executării unei operații mai complexe.

- fiecare operație de executat înseamnă o secvență unică de semnale de control generate de UC.

- fiecare operație se execută **secvențial**.

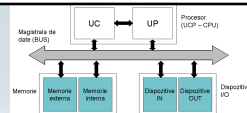
UC determină care operație elementară trebuie executată pe baza unui cod binar citit din memorie.

→ acesta este decodificat (recunoscut) de UC iar apoi este declanșată secvența tipică de semnale de control ce conduc ulterior la execuția operației dorite.



Operațiile se înlănțuie astfel:

operație de executat →  
cod binar (instrucțiune) →  
decodificare de către UC (recunoaștere) →  
declanșare secvență tipică de semnale de control →  
**se execută operația dorită**



## 1.2. Hardware - Software

După cum am menționat, pentru execuția unei anumite sarcini (job) sistemul de calcul are nevoie de o succesiune de instrucțiuni (coduri binare)

→ declanșează o secvență de operații elementare efectuate de blocurile constituente.

**program** = secvență de instrucțiuni ce sunt recunoscute de sistemul de calcul și pot fi executate de acesta.

→ sunt organizate logic și coerent după un anumit **algoritm**.

**algoritm** = un procedeu (inteligent) de combinare a unor operații standard în scopul realizării unei anumite prelucrări mai complexe a informației.

**hardware** = ansamblul structurii fizice a sistemului de calcul.



Ex. PC clasic



Ex. DSP – Digital Signal Processor

**software** = mulțimea programelor necesare sistemului de calcul pentru a îndeplini o serie de sarcini.

*Exemple:* sisteme de operare (Windows, Linux), procesoare de text (Word, WinEdt), medii de dezvoltare (Borland Builder, Matlab), antivirusi, prelucrare de imagini, etc.

> Funcționarea calculatorului trebuie privită prin prisma **dualității hardware – software**.

> Acestea nu își au rostul considerate separat:

- un sistem hardware fără sistem de operare nu are nici o valoare fiind inutilizabil. De asemenea un sistem hardware fără software adecvat este nerentabil.

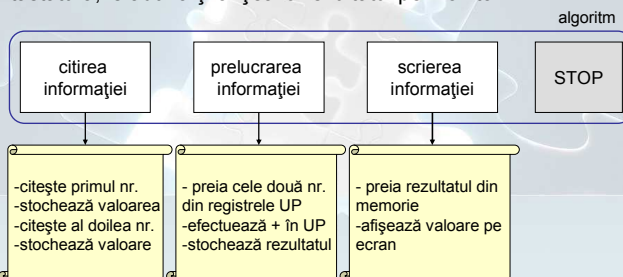
*Exemplu:* degeaba procesorul prelucrează date pe 64 biți dacă programele lucrează pe 32 biți,

- un pachet de programe nu-și are sensul fără sistemul fizic hardware (este mai folosit în bibliotecă).



O primă abordare a soluționării unei probleme de calcul folosind un sistem de calcul

*Enunț:* să se scrie un algoritm care citește două numere de la tastatură, le adună și afișează rezultatul pe monitor.



### 1.3. Limbaje de programare (generalități)

**Programul** este “intermediarul” dintre problema ce trebuie rezolvată, formulată în limbaj natural, și activitatea concretă a sistemului de calcul, formulată în limbaj mașină.

**Limbajul de programare** = limbaj *formal* în care se scrie programul. Ideal, cât mai intuitiv pentru utilizator dar și cât mai rapid.

*date:* obiectele programelor, valori numerice, valori textuale, date complexe, etc.

*instrucțiuni:* codifică operațiile ce trebuie executate de sistem, ex.: **printf** (scriere pe dispozitivul de ieșire) etc.



## Datele – obiectele programelor

- **date elementare**: a căror structură nu poate fi modificată de către utilizator.
  - > Acestea sunt de mai multe tipuri: *numerice* (întregi, reale), *logice* (booleene 0 sau 1), *alfanumerice* (caractere, text).
- **date structurate**: date complexe (grupuri de date elementare), ce poartă informație atât prin valoare cât și prin structură.
  - > Acestea sunt de mai multe tipuri: *tablouri*, *structuri*, *clase*.

1/9	1/9	1/9	structură "persoana" { Nume (alfanumeric) Vârstă (întreg) Greutate (real) ... }
1/9	1/9	1/9	
1/9	1/9	1/9	

## Datele – obiectele programelor (continuare)

> Datele sunt desemnate în cadrul programelor prin intermediul **variabilelor** și a constantelor (caz particular de variabilă).

**variabila** = fizic reprezintă o zonă de memorie ce găzduiește anumite date.

- variabilele sunt desemnate prin nume simbolice denumite și *identificatori*.
- variabilele sunt de un *anumit tip* ce indică tipul valorilor conținute de acestea precum și structura acestora (simplă, tablou, structură de date, etc.)

*Exemplu*: **int x**; → identificator="x", tip de date=întregi (int);  
la o anumită adresă se va aloca spațiu pentru valorile lui x.

## Instrucțiunile – componenta funcțională

Pot fi:

- **declarații**: prin care se definesc identificatorii și atributele variabilelor.

**int x** (de acum înainte "x" desemnează o variabilă întreagă)  
**float y=3.1** (de acum înainte "y" este real și are valoarea 3.1)

- **comenzi**: prin care se realizează prelucrarea efectivă a datelor conținute în variabile și constante.
  1. *comenzi elementare*: atribuirea de valori, citirea, scrierea, apelarea de funcții, etc.
  2. *comenzi structurate*: decizia, execuția în buclă, etc.

## Limbaje de programare

> Există o mare varietate de limbaje de programare. Acestea diferă unele de altele în funcție de următoarele aspecte:

- apropierea de limbajul mașină,
- problemele care le rezolvă,
- modul în care se stabilește asocierea nume simbolic – variabilă,
- facilitățile oferite utilizatorului,
- modul în care sunt apelate prelucrările

## Limbaje de programare – scurt istoric

~1947 primele limbaje de programare sunt limbajele mașină; gamă de instrucțiuni minimă, apropiată de hardware  
→ dificultatea de scriere a programelor complexe

<i>cod binar</i>	<i>cod hexazecimal</i>	
10001011	8B	} citire număr întreg în registrul AX
01000101	45	
00001010	0A	} citire număr întreg și adăunare în registrul AX
00000011	03	
01000101	45	
00010100	14	

**1954 FORTRAN** (John Backus IBM), programare de *nivel înalt*, permitea folosirea numelor de variabile, expresii complexe și proceduri (sub-programe).  
→ științific (FORMula TRANslating)

## Limbaje de programare – scurt istoric (continuare)

### 1954 FORTRAN

Revizuit în 1978 și 1990, încă folosit de comunitatea științifică datorită eficienței acestuia și a bibliotecilor de funcții de prelucrare disponibile (cel mai longeviv limbaj).

```
READ INPUT TAPE 5, 501, IA, IB, IC  
501 FORMAT (3I5)
```

```
IF (IA) 777, 777, 701  
701 IF (IB) 777, 777, 702  
702 IF (IC) 777, 777, 703  
703 IF (IA+IB-IC) 777,777,704  
704 IF (IA+IC-IB) 777,777,705  
705 IF (IB+IC-IA) 777,777,799  
777 STOP 1
```

### Limbaje de programare – scurt istoric (continuare)

**1958** ALGOL: folosire restricționată (limbaj sub licența), concurență FORTRAN.

**1960** COBOL destinat aplicațiilor de gestiune, sintaxa cât mai apropiată de limba engleză.

ADD YEARS TO AGE → age = age + years

**1963** BASIC (Beginner's All-purpose Symbolic Instruction Code), în principal cu scop educativ.

> În ciuda diversității reduse de instrucțiuni devine foarte popular datorită ușurinței utilizării acestuia.

> Programe nestructurate → mentenanță dificilă.

### Limbaje de programare – scurt istoric (continuare)

**1970** Pascal (Niklaus Wirth) dezvoltat în scopul predării programării structurate și modulare.

Preia punctele forte ale limbajelor COBOL, FORTRAN și ALGOL → limbaj "elegant", simplu, înlocuiește BASIC ca limbaj de inițiere.

```
program Hello(output);  
var  
  a:string;  
begin  
  write('introduceți numele: ');  
  readln(a);  
  writeln('salut ', a);  
end.
```

### Limbaje de programare – scurt istoric (continuare)

**1970** C (Dennis Ritchie, laboratoarele Bell) dezvoltat în scopul programării sistemului de operare UNIX.

> Limbaj de programare "puternic". Datorită folosirii pointerilor permite apropierea de limbajul mașină precum și accesul la dispozitivele hardware ale sistemului, rămânând totuși un limbaj de programare de nivel înalt.

**1980** Smalltalk-80 (inițial Alan Kay 1969, ulterior Xerox) motivat de necesitatea de programe tot mai complexe; propune o nouă direcție de programare = *programarea orientată pe obiecte*.

**1985** C++ (laboratoarele Bell) extensie obiect orientată a limbajului C, instrucțiuni noi, programare mai eficientă, viteză de lucru crescută.

### Limbaje de programare – scurt istoric (continuare)

**1980-1990** limbaje interpretate sau semi-interpretate motivate de dezvoltarea Web (dezvoltare de pagini Web dinamice, aplicații client-server, etc.)

*Exemple:* Perl (Larry Wall, 1987), Tcl (John Ousterhout, 1988), Python (Guido van Rossum, 1990), PHP și Java (Sun Microsystems, 1996) ...

**1995** Common-LISP: primul limbaj orientat obiect standardizat de ANSI - American National Standards Institute.

**2000** C# (Microsoft): permite folosirea simultană a mai multe tipuri de programare (multi-paradigm).

...

### Paradigme de programare (moduri de programare)

- **Programare imperativă:** în care calcul înseamnă o secvență de comenzi (FORTRAN, C, Pascal).

*Exemplu:* citește X, citește Y, calculează X\*Y, pune rezultatul în variabila Z.

- **Programare funcțională:** în care calcul înseamnă evaluarea unor funcții în sensul matematic (Lisp, Python). Programele sunt grupuri de funcții (sub-programe).

- **Programare obiect orientată:** definirea de obiecte care interacționează între acestea prin intermediul mesajelor (C++, Java).

### Paradigme de programare (continuare)


- **Programare logică:** în care calcul înseamnă o serie de declarații logice (Prolog).

*Exemplu:* Cezar este om, Toți oamenii sunt muritori.  
→ Cezar este muritor.

- **Programare concurentă:** în care calculul este divizat în mai multe sarcini ce apoi sunt executate în paralel.

**Notă:** Un anumit limbaj de programare poate oferi mai multe moduri de programare, acestea nu sunt exclusive.

*Exemplu:* C++, programare funcțională, obiect orientată, concurentă, etc.



## Sfârșitul Cursului 1