



PROGRAMARE PROCEDURALĂ

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

Secția Informatică, anul I,

2017-2018

Cursul 1

Cuprinsul cursului de azi

1. Bun venit la Facultatea de Matematică și Informatică!
2. Prezentarea cursului de Programare Procedurală
3. Primul curs

Bun venit la FMI!

Prezentarea facultății:

- Site-ul facultății: <http://fmi.unibuc.ro/>
- Prezentare generală a facultății:
[http://fmi.unibuc.ro/ro/pdf/2017/admitere/licenta/
Brosura Licenta Facultatea de Matematica si Informatica-2017-
romana.pdf](http://fmi.unibuc.ro/ro/pdf/2017/admitere/licenta/Brosura_Licenta_Facultatea_de_Matematica_si_Informatica-2017-romana.pdf)
- Ghidul bobocului: <http://boboc.as-mi.ro/>

Lucruri bine de știut de studenți

- 3 ani = 6 semestre = $5 \times 14 + 1 \times 10 = 80$ săptămâni
- 6 sesiuni = $6 \times 3 = 18$ săptămâni
- dacă vă luați toate examenele deveniți absolvent
- vă luați și examenul final de licență și deveniți licențiat
- timpul trece repede: o săptămână $\approx 1\%$
- **condiții de trecere dintr-un an în altul:**
 - din anul 1 în anul 2: cel puțin 30 de credite
 - din anul 2 în anul 3: maxim 4 restanțe din primii 2 ani, cu maxim o restanță din anul 1 (regula actuală)

Planuri de învățământ

Anul 1

(Sem.I - 14 săptămâni; Sem.II - 14 săptămâni)

Total Credite: 60

Nr. crt.	Discipline obligatorii	Semestrul I					Semestrul II				
		C	S	L	FE	Crd.	C	S	L	FE	Crd.
1.	Algebră	2	1	—	E	4	2	1	—	E	4
2.	Analiză	2	1	—	E	4	2	1	—	E	4
3.	Logică matematică și computațională	2	2	—	E	4	—	—	—		
4.	Programare procedurală	2	1	2	V	6	—	—	—		
5.	Arhitectura sistemelor de calcul	2	1	1	V	6	—	—	—		
6.	Algoritmi și structuri de date	2	1	2	E	6	—	—	—		
7.	Geometrie	—	—	—			2	1	—	E	4
8.	Limbaje formale și automate	—	—	—			2	1	1	V	6
9.	Programare orientată pe obiecte	—	—	—			2	1	2	E	6
10.	Algoritmica grafurilor	—	—	—			2	1	1	V	6
	Total	12	7	5	4E+2V	30	12	6	4	4E+2V	30

Planuri de învățământ

Anul 2

(Sem.I - 14 săptămâni; Sem.II - 14 săptămâni)

Total Credite: 60

Nr. crt.	Discipline obligatorii	Semestrul I					Semestrul II				
		C	S	L	FE	Crd.	C	S	L	FE	Crd.
1.	Geometrie computatională	2	—	1	E	5	—	—	—		
2.	Calculabilitate și complexitate	2	2	—	E	5	—	—	—		
3.	Tehnici avansate de programare	2	1	2	V	5	—	—	—		
4.	Probabilitati si statistica	2	1	1	E	5	—	—	—		
5.	Tehnici Web	2	—	2	V	5	—	—	—		
6.	Sisteme de operare	2	—	2	E	5	—	—	—		
7.	Rețele de calculatoare	—	—	—			2	—	2	V	5
8.	Inteligentă artificială	—	—	—			2	1	2	E	5
9.	Baze de date	—	—	—			2	—	3	E	5
10.	Programare avansata pe obiecte	—	—	—			2	—	2	V	5
11.	Programare logică	—	—	—			2	—	2	E	5
12.	Metode de dezvoltare software	—	—	—			3	—	2	E	5
Total		12	4	8	4E+2V	30	13	1	13	4E+2V	30

Planuri de învățământ

Anul 3

(Sem.I - 14 săptămâni; Sem.II - 10 săptămâni)

Total Credite: 60

Informatica la FMI

Principalele direcții de studiu sunt:

- structuri de date și algoritmi
- limbaje de programare (C, C++, Java, C#, limbaje non-procedurale)
- baze de date (SQL, PL/SQL, Oracle),
- dezvoltare de aplicații web (HTML, CSS, Java Script, XML, JSP, ASP.NET)
- administrare de rețele (Unix, Windows)

Regulamente UB & FMI

- regulament privind activitatea studenților la UB:
http://fmi.unibuc.ro/ro/pdf/2015/consiliu/UB_Regulament_studenti_2015.pdf

- regulament de etică și profesionalism la FMI:

http://fmi.unibuc.ro/ro/pdf/2015/consiliu/Regulament_etica_FMI.pdf

Se consideră **incident minor** cazul în care un student/ o studentă:

- a. preia codul sursă/ rezolvarea unei teme de la un coleg/ o colegă și pretinde că este rezultatul efortului propriu;

Se consideră **incident major** cazul în care un student/ o studentă:

- a. copiază la examene de orice tip;

- **3 incidente minore = un incident major = exmatriculare**

„copiul” a primit
de materialul din „copiul” și găsit
în locul ocupat deu așa către. În
continu foarte scură de o altă
și diferit, cu o altă cenușă).
în urma evenimentului de pe incident
major.

Exmatriculare pentru grădiniță
în drept de invazie la adresa
conform consiliului PMI 12.03.2016

Blera

Cuprinsul cursului de azi

1. Bun venit la Facultatea de Matematică și Informatică!
2. Prezentarea cursului de Programare Procedurală
3. Primul curs

Prezentarea cursului de PP

- Utilitatea cursului de PP
- Structura primului semestru
- Orar
- Materiale
- Obiectivele cursului
- Programa cursului
- Bibliografie și resurse online
- Regulament de notare si evaluare
- Notele din anii trecuți (Mate și Info)

Utilitatea cursului de PP

- PP = paradigma de programare bazată pe conceptul de apel de procedură/funcție/rutină/ subrutină. Un program este privit ca o multime ierarhică de funcții care manipulează datele.
- vom studia limbajul C = limbaj fundamental de programare (1970), exponent al programării procedurale. Alte limbaje (C++, Java, PHP, Python) împrumută multe din caracteristicile limbajului C.

De ce C?

<http://www.tiobe.com/tiobe-index/>

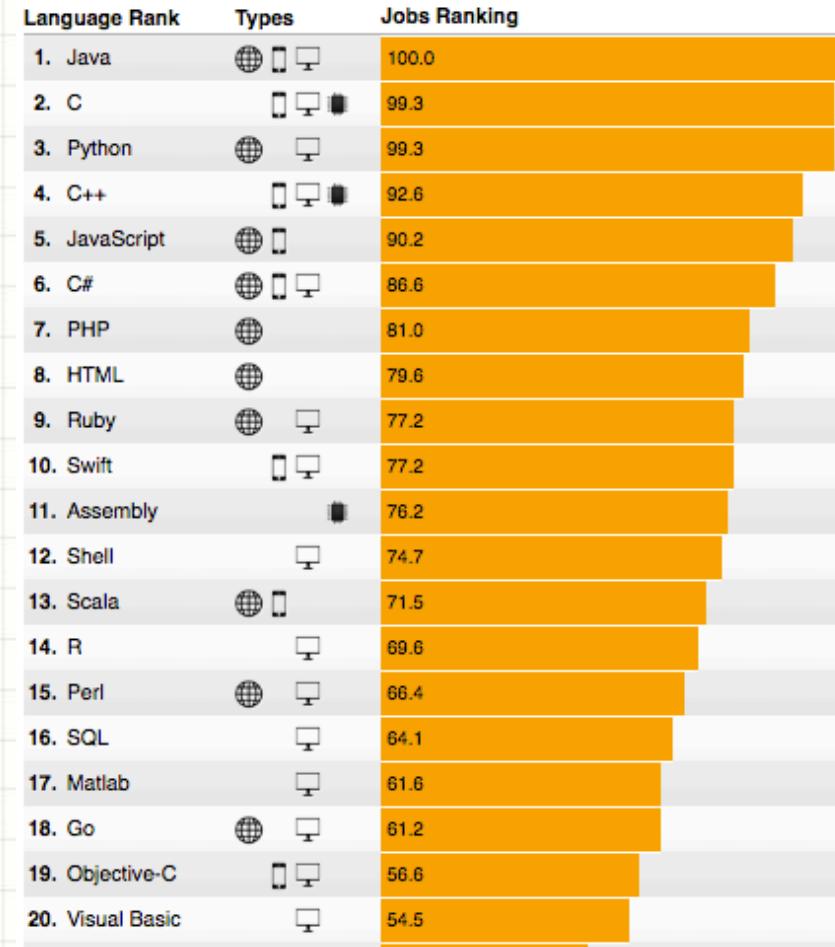
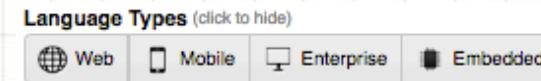
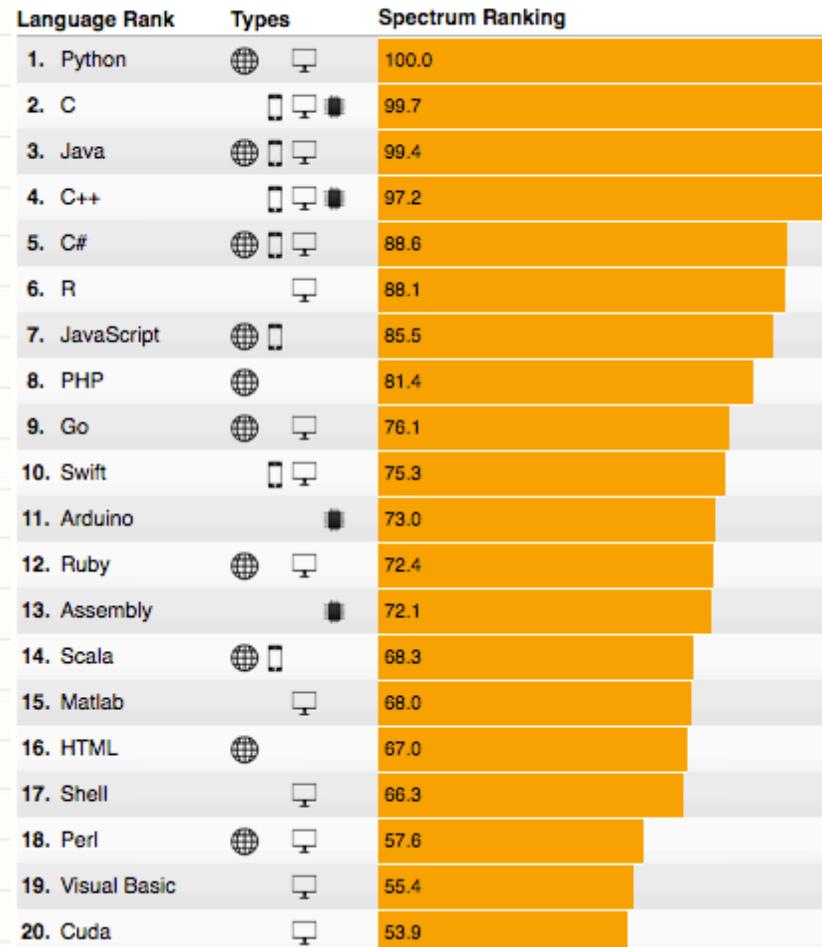
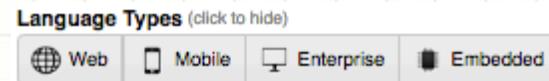
The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

Sep 2017	Sep 2016	Change	Programming Language	Ratings	Change
1	1		Java	12.687%	-5.55%
2	2		C	7.382%	-3.57%
3	3		C++	5.565%	-1.09%
4	4		C#	4.779%	-0.71%
5	5		Python	2.983%	-1.32%
6	7	▲	PHP	2.210%	-0.64%
7	6	▼	JavaScript	2.017%	-0.91%
8	9	▲	Visual Basic .NET	1.982%	-0.36%
9	10	▲	Perl	1.952%	-0.38%
10	12	▲	Ruby	1.933%	-0.03%
11	18	▲	R	1.816%	+0.13%
12	11	▼	Delphi/Object Pascal	1.782%	-0.39%
13	13		Swift	1.765%	-0.17%
14	17	▲	Visual Basic	1.751%	-0.01%
15	8	▼	Assembly language	1.639%	-0.78%
16	15	▼	MATLAB	1.630%	-0.20%
17	19	▲	Go	1.567%	-0.06%
18	14	▼	Objective-C	1.509%	-0.34%
19	20	▲	PL/SQL	1.484%	+0.04%
20	26	▲	Scratch	1.376%	+0.54%

De ce C?

<http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017>

"Rankings are created by weighting and combining 12 metrics from 10 sources. We offer preset weightings—the default is our IEEE Spectrum ranking—but there are presets for those interested in what's trending or most looked for by employers."



Structura primului semestru

http://unibuc.ro/n/studii/calendar_academic.php

Studii universitare de licență și masterat

Semestrul	Anul de studii	PERIOADA	ACTIVITATEA
I	I, II, III, IV MASTER	02.10.2017 – 20.12.2017	Activitate didactică
		21.12.2017 – 03.01.2018	Vacanță de iarnă
		04.01.2018 – 20.01.2018	Activitate didactică
		21.01.2018 – 11.02.2018	Sesiune de examene
		12.02.2018 – 18.02.2018	Vacanță intersemestrială
		25.01.2018 – 15.02.2018	Sesiune de licență, disertație

- 13 cursuri de PP (30 noiembrie fără curs): 10 cursuri în 2017 + 3 cursuri în 2018 (**cursul de pe 4 ianuarie se face!!!**)

Orar

Alexe Bogdan

Universitatea din Bucuresti, Facultatea de Matematica si Informatica, str. Academiei 14, Bucuresti

	8 8:00 - 8:50	9 9:00 - 9:50	10 10:00 - 10:50	11 11:00 - 11:50	12 12:00 - 12:50	13 13:00 - 13:50	14 14:00 - 14:50	15 15:00 - 15:50
Jo	ProgProced 133 3	ProgProced 131/132/133/134/135 3	2(Pompeiu)				ProgProced 134 3	ProgProced 132 3

- Curs – 2 ore/săptămână
- Laborator – 2 ore/săptămână (cu semi-grupa)
- Seminar – 2 ore/ 2 săptămâni (cu grupa): săptămâna 3-7 octombrie e săptămână impară

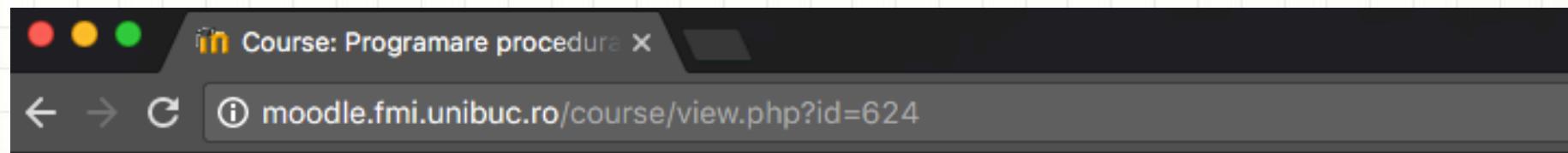
Orar

Formarea semigrupelor de la laborator:

- aveți libertatea să vă împărtiți cum vreți;
- dorim ca semi-grupele să fie echilibrate ca număr;
- dacă nu ajungeți la un consens formăm semigrupele după ordinea alfabetică (prima jumătate a catalogului = prima semi-grupă, a doua jumătate = a doua semi-grupă);
- semi-grupe definite începând cu săptămâna 3;
- nu permitem apoi să vă transferați de la o semi-grupă la alta.

Materiale

- moodle.fmi.unibuc.ro



Programare procedurala

Home ► Courses ► Zi ► Departament Informatica ► Alexe Bogdan ► ProgProced1

Navigation -||-

- Home
- Site pages
- ▼ Courses
 - Zi
 - ▼ Departament Informatica
 - Adam Mircea
 - ▼ Alexe Bogdan
 - InteligArtificiala2
 - ProgProced1

Topic outline

-  News forum
- 1
- 2
- 3

Obiectivele cursului

1. Formarea deprinderilor de **programare structurată (modularizare)** în limbaje de programare clasice și moderne (descompunerea unei probleme complexe în subprobleme relativ simple și independente);
2. Însușirea caracteristicilor **limbajului C**: alocarea memoriei, lucrul cu pointerii, lucrul cu fișierele, programarea generică. Vrem să știți să codați în C, să vă dați seama ce face un cod scris de altcineva, să depanați un cod în C;
3. Dezvoltarea unei **gândiri algoritmice + abilitate de programare** - foarte utile în rezolvarea diverselor probleme cu care vă veți întâlni în facultate sau în viața reală.

Programa cursului

□ Introducere

- Algoritmi
- Limbaje de programare.
- Introducere în limbajul C. Structura unui program C.

□ Fundamentele limbajului C

- Tipuri de date fundamentale. Variabile. Constante. Operatori. Expresii. Conversii.
- Tipuri derivate de date: tablouri, siruri de caractere, structuri, uniuni, câmpuri de biți, enumerări, pointeri
- Instrucțiuni de control
- Directive de preprocesare. Macrodefiniții.
- Funcții de citire/scriere.
- Etapele realizării unui program C.

□ Fișiere text

- Funcții specifice de manipulare.

□ Funcții (1)

- Declarație și definire. Apel. Metode de transmitere a parametrilor. Pointeri la funcții.

□ Tablouri și pointeri

- Legătura dintre tablouri și pointeri
- Aritmetică pointerilor
- Alocarea dinamică a memoriei
- Clase de memorare

□ Siruri de caractere

- Funcții specifice de manipulare.

□ Fișiere binare

- Funcții specifice de manipulare.

□ Structuri de date complexe și autoreferite

- Definire și utilizare

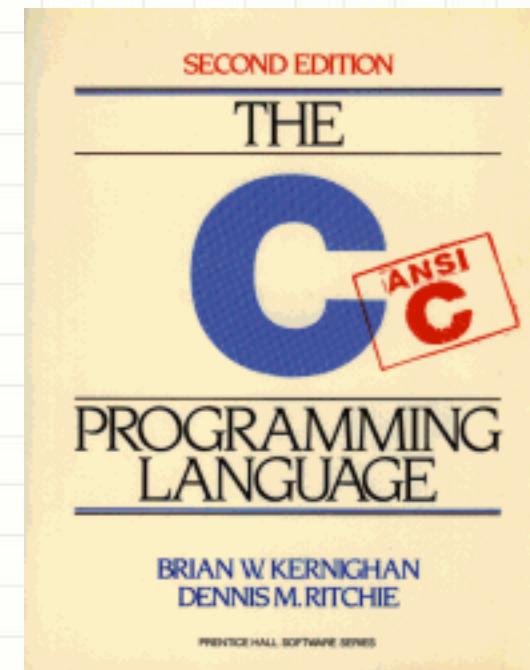
□ Funcții (2)

- Funcții cu număr variabil de argumente.
- Preluarea argumentelor funcției main din linia de comandă.
- Programare generică.

□ Recursivitate

Bibliografie

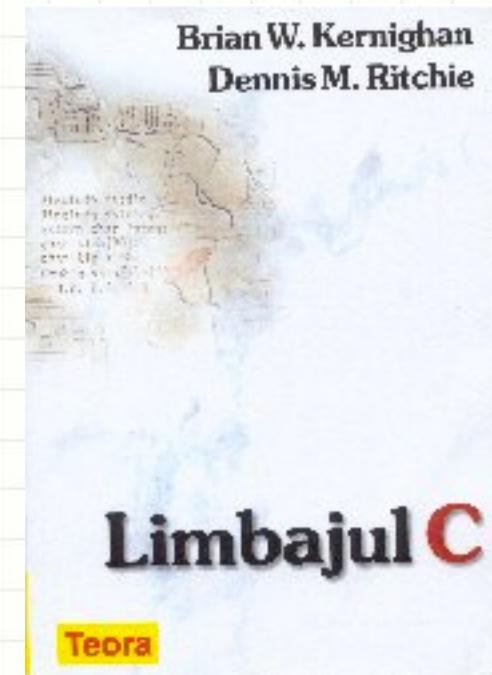
1. Kernighan & Ritchie: The C programming language
<http://zanasi.chem.unisa.it/download/C.pdf>



Bibliografie

1. Kernighan & Ritchie: The C programming language
<http://zanasi.chem.unisa.it/download/C.pdf>

2. Kernighan & Ritchie: Limbajul C
Editura Teora, 2003

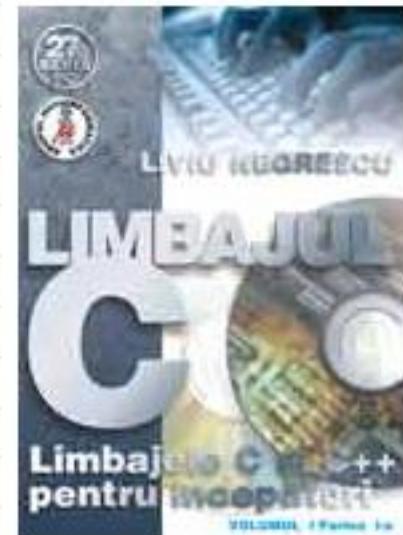


Bibliografie

1. Kernighan & Ritchie: The C programming language
<http://zanasi.chem.unisa.it/download/C.pdf>

2. Kernighan & Ritchie: Limbajul C
Editura Teora, 2003

3. Liviu Negrescu: Limbajele C si C++ pentru începători,
volumul 1, partea I si II (Limbajul C)
Editura Albastra, 2001



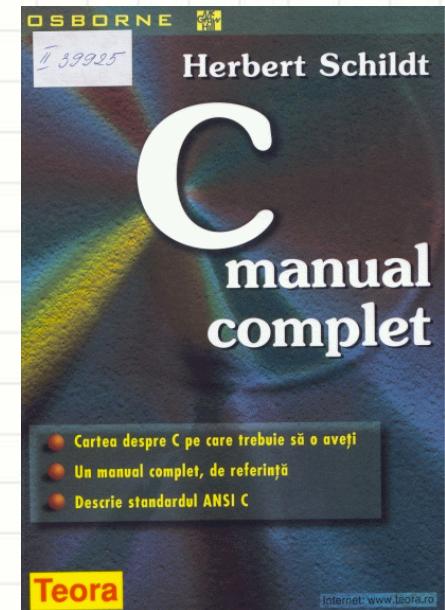
Bibliografie

1. Kernighan & Ritchie: The C programming language
<http://zanasi.chem.unisa.it/download/C.pdf>

2. Kernighan & Ritchie: Limbajul C
Editura Teora, 2003

3. Liviu Negrescu: Limbajele C si C++ pentru începători,
volumul 1, partea I si II (Limbajul C)
Editura Albastra, 2001

4. Herbert Schildt: C, manual complet.
Editura Teora, 2000?



Bibliografie

1. Kernighan & Ritchie: The C programming language

<http://zanasi.chem.unisa.it/download/C.pdf>



Sixth Edition

2. Kernighan & Ritchie: Limbajul C

Editura Teora, 2003

3. Liviu Negrescu: Limbajele C si C++ pentru începători,

volumul 1, partea I si II (Limbajul C)

Editura Albastra, 2001

4. Herbert Schildt: C, manual complet.

Editura Teora, 2000?

C Primer Plus



5. Stephan Prata: C primer plus, 6th Edition

[https://vk.com/doc190970339_430409589?
hash=2d2b4245bd65b25e27&dl=cd4e96f98aeddd5c1e](https://vk.com/doc190970339_430409589?hash=2d2b4245bd65b25e27&dl=cd4e96f98aeddd5c1e)

Resurse online

- Cursuri online:
 - căutare după cuvinte cheie “online lectures C programming”
 - universități americane de prestigiu au conținutul cursurilor gratuit:
 - Stanford:
<https://see.stanford.edu/Course/CS107> (online video lectures)
 - MIT:
<https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-087-practical-programming-in-c-january-iap-2010/lecture-notes/>
- Site-uri cu probleme de programare pentru concursuri
 - www.infoarena.ro
 - www.acm.ro
 - www.topcoder.com

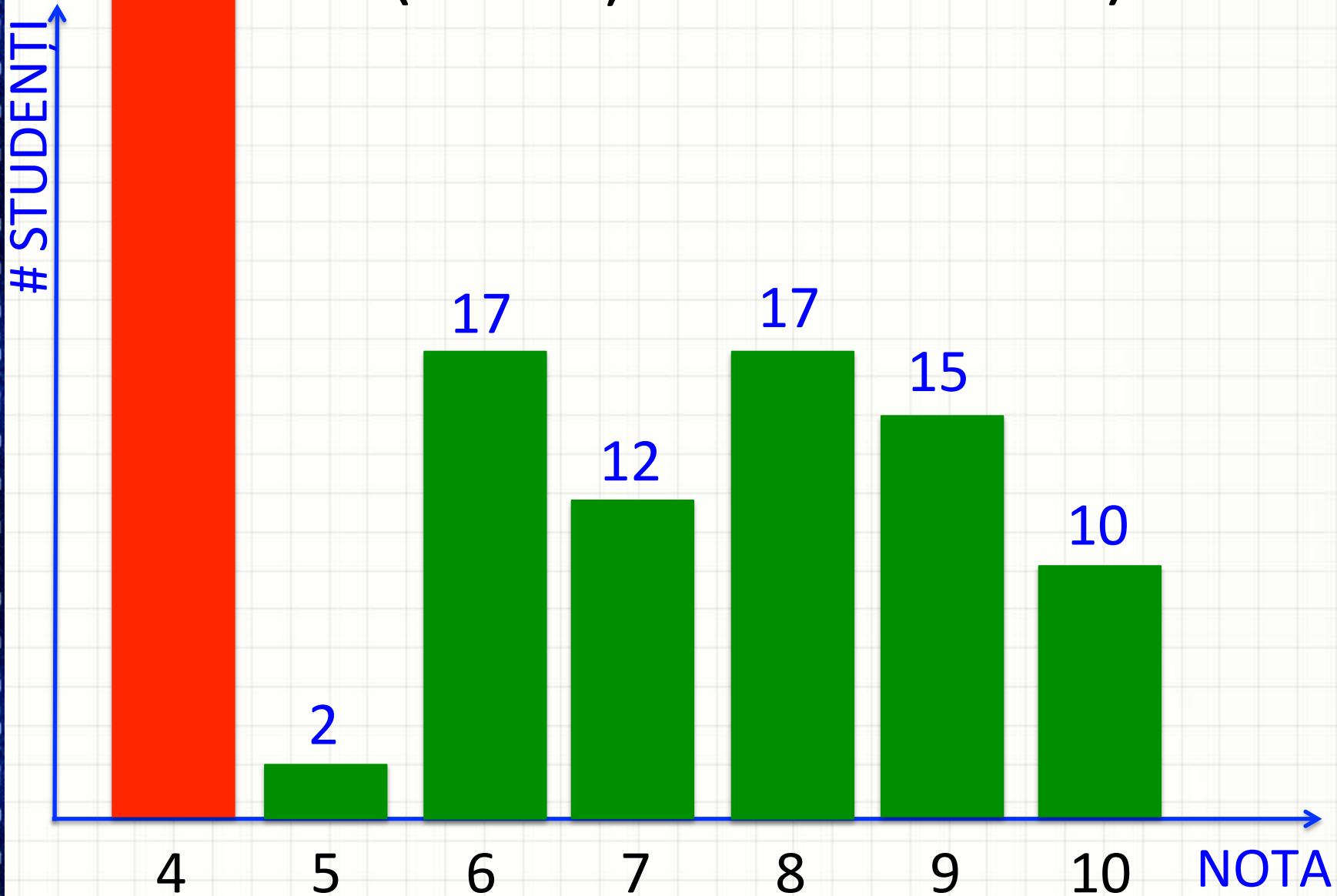
Regulament de evaluare și notare

Nu este înr-o formă finală detaliată, trebuie să ne întâlnim cu asistenții de laborator. Îl definitivăm și vi-l prezint la cursul al doilea.

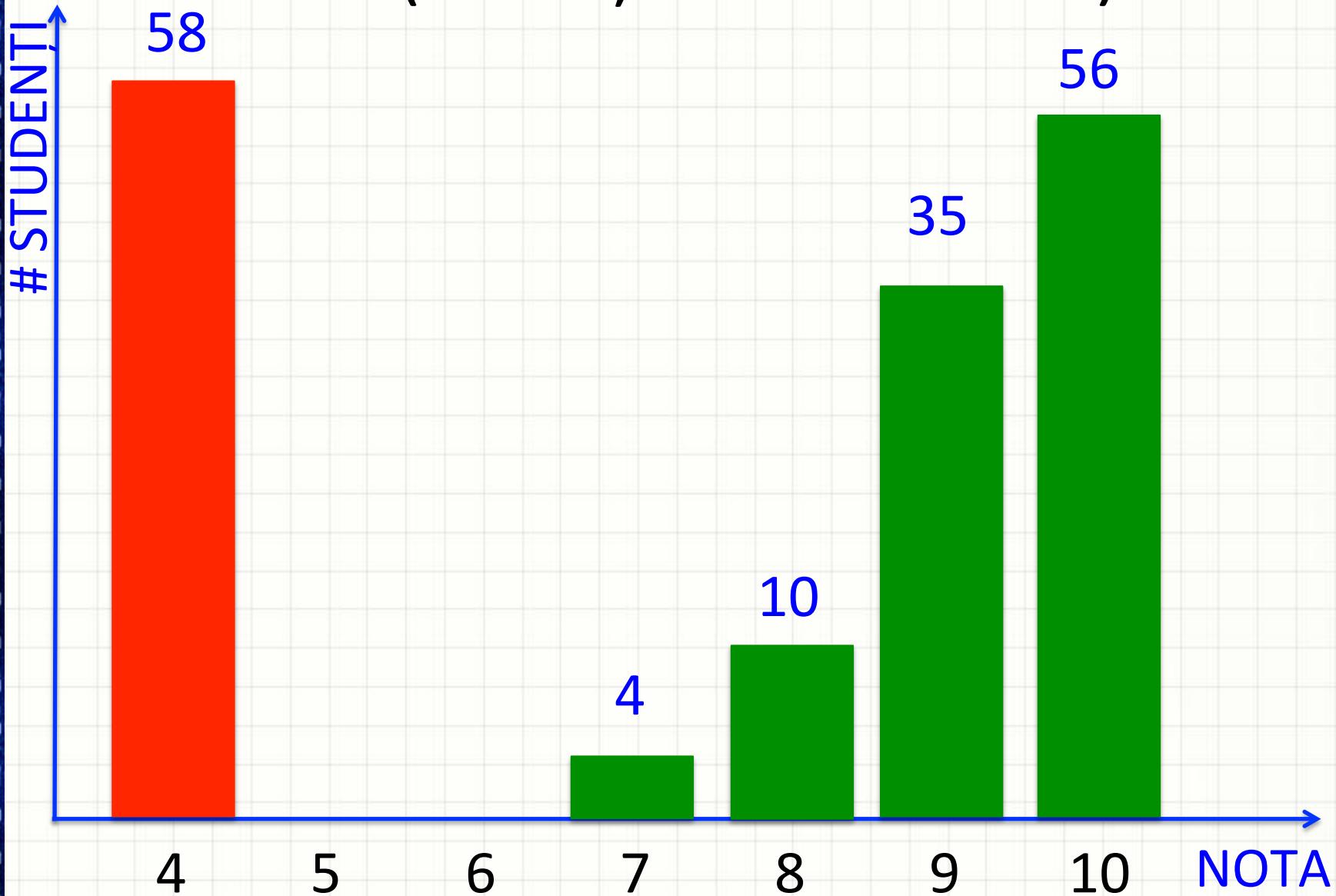
$$Nota = \min (10, Curs + Laborator + Seminar)$$

6p 4p 1p

Notele de acum 2 ani - ianuarie (la secția Matematică)



Notele de anul trecut - ianuarie (la secția Informatică)



Mulțumiri pentru materiale/slides-uri:

- Anca Dobrovăț (FMI)
- Radu Boriga (FMI)
- Cristina Dăscălescu (FMI)
- Grigore Albeanu (FMI)
- Vlad Posea (Politehnică București)
- Traian Rebedea (Politehnică București)
- Kinga Marton (Politehnică Cluj)
- Ion Giosan (Politehnică Cluj)
- mulți alții ...

Cuprinsul cursului de azi

1. Bun venit la Facultatea de Matematică și Informatică!
2. Prezentarea cursului de Programare Procedurală
3. Primul curs

Programa cursului

Introducere

Algoritmi

Limbaje de programare.

Introducere în limbajul C. Structura unui program C.

Fundamentele limbajului C

- Tipuri de date fundamentale. Variabile. Constante. Operatori. Expresii. Conversii.
- Tipuri derivate de date: tablouri, siruri de caractere, structuri, uniuni, campuri de biti, enumerari, pointeri
- Instrucțiuni de control
- Directive de preprocessare. Macrodefiniții.
- Funcții de citire/scriere.
- Etapele realizării unui program C.

Fișiere text

- Funcții specifice de manipulare.

Funcții (1)

- Declarare și definire. Apel. Metode de trasmisare a parametrilor. Pointeri la funcții.

Tablouri și pointeri

- Legătura dintre tablouri și pointeri
- Aritmetică pointerilor
- Alocarea dinamică a memoriei
- Clase de memorare

Siruri de caractere

- Funcții specifice de manipulare.

Fișiere binare

- Funcții specifice de manipulare.

Structuri de date complexe și autoreferite

- Definire și utilizare

Funcții (2)

- Funcții cu număr variabil de argumente.
- Preluarea argumentelor funcției main din linia de comandă.
- Programare generică.

Recursivitate

Cursul 1:

1. Algoritmi

2. Limbaje de programare

3. Introducere în limbajul C. Structura unui program C.

Algoritmi

Rezolvarea oricărei probleme implică mai multe etape:

1. Analiza problemei
2. Găsirea soluției [optime]
3. Elaborarea algoritmului
4. Implementarea algoritmului într-un limbaj de programare
5. Verificarea corectitudinii algoritmului propus
6. Analiza complexității

Algoritmi

Algoritm = o succesiune finită, ordonată și bine definită (exprimată clar și precis) de operații executabile (instrucțiuni, pași) care constituie o metodă corectă de rezolvare a unei probleme pornind dintr-o stare inițială, folosind datele disponibile și ajungând în starea finală dorită.

Exemplu: algoritmul lui Euclid pentru determinarea celui mai mare divizor comun a două numere naturale (scăderi repetitive, resturi)

$$1599 = 650 \times 2 + 299$$

$$650 = 299 \times 2 + 52$$

$$299 = 52 \times 5 + 39$$

$$52 = 39 \times 1 + 13$$

$$39 = 13 \times 3 + 0$$

$$\text{cmmdc}(1599, 650) = 13$$

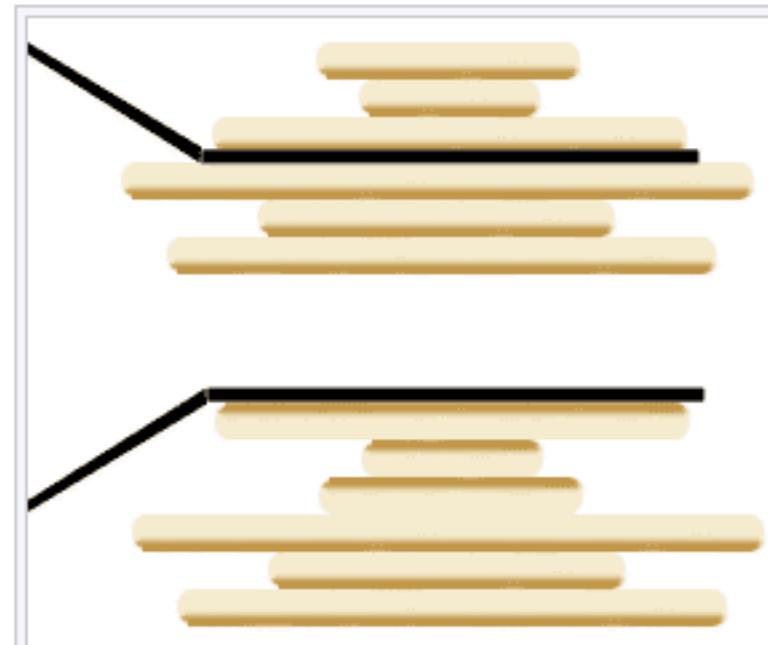
1599

Algoritmi - exemplu

Pancake sorting

From Wikipedia, the free encyclopedia

Pancake sorting is the colloquial term for the mathematical problem of sorting a disordered stack of pancakes in order of size when a [spatula](#) can be inserted at any point in the stack and used to flip all pancakes above it. A *pancake number* is the minimum number of flips required for a given number of pancakes. In this form, the problem was first discussed by [American geometer Jacob E. Goodman](#).^[1] It is a variation of the sorting problem in which the only allowed operation is to reverse the elements of some *prefix* of the sequence. Unlike a traditional



Demonstration of the primary operation. The spatula is flipping over the top three pancakes, with the result seen below. In the burnt pancake problem, their top sides would now be burnt instead of their bottom sides.

Algoritmi - exemplu

Pancake sorting

Universitatea din Bucureşti
Facultatea de Matematică și Informatică

14.07.2017

IV. Informatică.

Fie n un număr natural nenul. Fie v un vector cu n poziții numerotate de la 1 la n și elemente numere naturale diferite, de la 1 la n , într-o ordine oarecare. Pentru i și j numere naturale între 1 și n , numim $\text{FLIP}(n, v, i, j)$ operația care inversează ordinea elementelor din v situate pe pozițiile de la i la j .

- a) Să se scrie în limbaj de programare o procedură (sau funcție) care implementează operația $\text{FLIP}(n, v, i, j)$.
- b) Să se scrie un program care sortează crescător vectorul v , folosind pentru schimbarea ordinii elementelor în v doar operația $\text{FLIP}(n, v, 1, k)$, cu k de la 2 la n .
- c) Considerăm că n este o putere a lui 2 ($n = 2^m$, cu m număr natural nenul) și vectorul v are proprietatea că pentru orice i de la 1 la m și orice j de la 1 la 2^{m-i} , există k de la 1 la 2^{m-i} , astfel încât pe pozițiile din v de la $2^i(j-1)+1$ la $2^i j$ se află numerele naturale de la $2^i(k-1)+1$ la $2^i k$, într-o ordine oarecare. Să se scrie un program care sortează crescător vectorul v , folosind pentru schimbarea ordinii elementelor în v doar operația $\text{FLIP}(n, v, 2^i(j-1)+1, 2^i j)$, cu i de la 1 la m și j de la 1 la 2^{m-i} , printr-un algoritm mai eficient decât cel implementat la punctul b), care se bazează pe proprietatea vectorului v .



Reprezentarea algoritmilor

1. Pseudocod/ limbaj natural
2. Schemă logică
3. Program într-un limbaj de programare

Pseudocod

- limbaj natural structurat exprimat formal
- fiecare pas al algoritmului este reprezentat de o linie separată, ca o propoziție
- acțiuni (verbe) aplicate unor date (substantive)
- indentarea poate reda ierarhia instrucțiunilor

*Algoritmul lui Euclid
prin scăderi repetate*
cât timp $B > 0$
dacă $A > B$
 $A = A - B;$
altfel
 $B = B - A;$
afișează A

Schemă logică

- alăturare de simboluri vizuale care desemnează fluxul logic al pașilor



Bloc de instrucțiuni



Structura alternativă



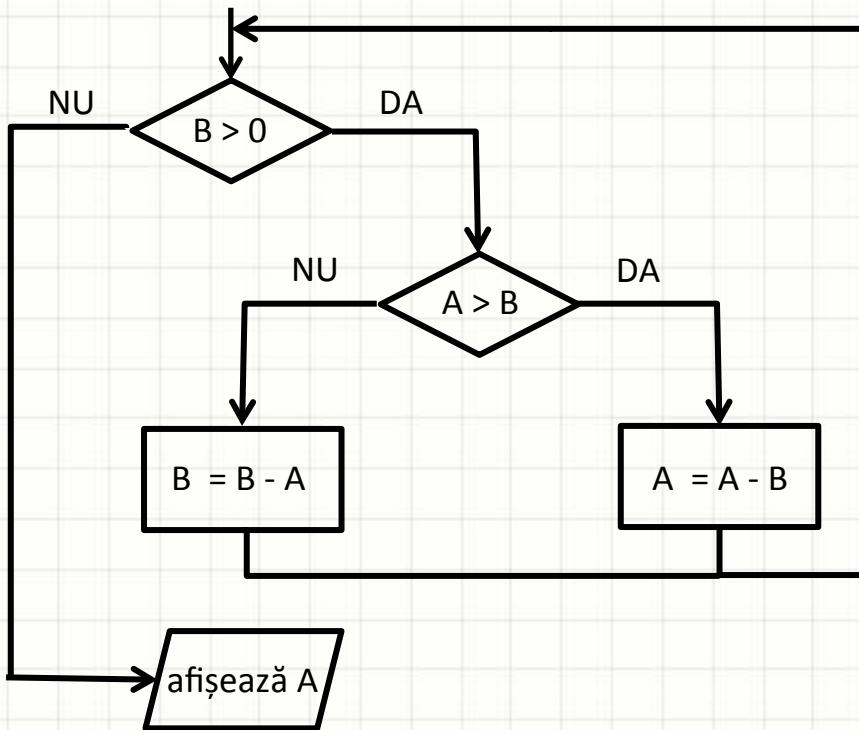
Direcția fluxului



Operația de intrare/ieșire

Schemă logică

- alăturare de simboluri vizuale care desemnează fluxul logic al pașilor



*Algoritmul lui Euclid
prin scăderi repetitive*
cât timp $B > 0$
dacă $A > B$
 $A = A - B;$
altfel
 $B = B - A;$
afișează A

Cursul 1:

1. Algoritmi

2. Limbaje de programare

3. Introducere în limbajul C. Structura unui program C.

Limbaje de programare

Rezolvarea oricărei probleme implică mai multe etape:

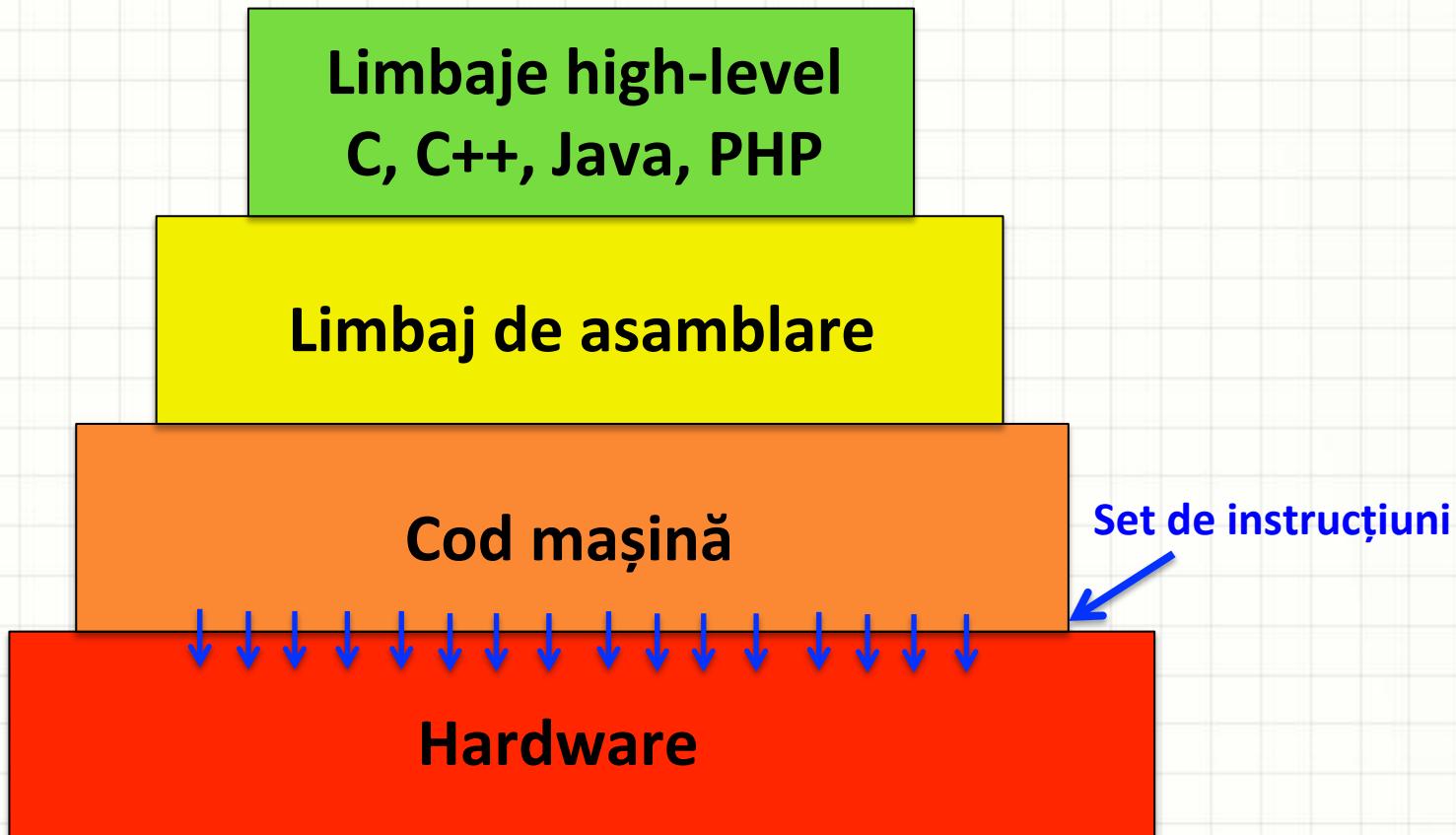
1. Analiza problemei
2. Găsirea soluției [optime]
3. Elaborarea algoritmului
4. Implementarea algoritmului într-un limbaj de programare
5. Verificarea corectitudinii algoritmului propus
6. Analiza complexității

Limbaj de programare

- limbaj artificial cu sintaxă și semantică bine definite
- pune la dispoziția programatorilor construcții sintactice prin care sunt specificate succesiunea de operații/instrucțiuni elementare (pe care un calculator le poate executa) asociate algoritmului de rezolvare a unei probleme;
- este necesară cunoașterea setului de operații/instrucțiuni elementare al calculatorului la care ne referim.
- **limbaj mașină** = limbajul nativ al unui calculator (mașină)

Limbaje low-level și high-level

- dată de apropierea unui limbaj de limbajul nativ al calculatorului (limbaj mașină = cod mașină)



Limbaje low-level (de nivel scăzut)

Limbaj mașină

- limbajul nativ al unui calculator (mașină);
 - şabloane de numere binare (reprezintă modul binar de codificare a instrucţiunilor şi datelor în memorie)
 - depinde de arhitectura sistemului

```
00000000010100001000000000000011000  
0000000000000110000001100000100001  
1000110001100010000000000000000000  
10001100111100100000000000000000100  
10101100111100100000000000000000000  
10101100011000100000000000000000100  
0000001111100000000000000000000000100
```

Instructiuni în limbaj masină

Limbaj de asamblare

- În loc de cod mașină folosește o desemnare simbolică a elementelor programului (instructiuni, date)
 - 01011011 = ADD, 01011100 = SUB

```
swap:  
    muli $2, $5,4  
    add $2, $4,$2  
    lw   $15, 0($2)  
    lw   $16, 4($2)  
    sw   $16, 0($2)  
    sw   $15, 4($2)  
    jr   $31
```

Instructiuni în
limbaj de asamblare

Limbaje high-level (de nivel înalt)

- cuprind mecanisme de exprimare apropriate de limbajul natural;
 - folosesc verbe pentru a desemna acțiuni (**do, repeat, read, write, continue, switch, call, goto**, etc.), conjunctii (**if, while**), adverbe (**then, else**), mecanisme de declarare si definire.

```
swap(int v[], int k)
{int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

Instructiuni în limbajul C

```
swap:  
    muli $2, $5,4  
    add $2, $4,$2  
    lw   $15, 0($2)  
    lw   $16, 4($2)  
    sw   $16, 0($2)  
    sw   $15, 4($2)  
    jr   $31
```

Instrucțiuni în
limbaj de asamblare

```
0000000001010000100000000000000011000  
00000000000011000000011000000100001  
10001100011000100000000000000000000000  
100011001111001000000000000000000000000  
1010110011110010000000000000000000000000  
1010110001100010000000000000000000000000  
00000001111100000000000000000000000000000
```

Instructiuni în limbaj mașină

Limbaje high-level (de nivel înalt)

- au o descriere sintactică și semantică bine definită
- descurajează greșelile de programare
- independente de procesor (pentru asigurarea portabilității codului)
- independente de sistemul de operare (pentru a permite realizarea de software multi-platforma)
- codul sursă se convertește în cod mașină folosind compilatoare sau interpretoare

Paradigme de programare

A: PARADIGMA PROGRAMARII PROCEDURALE SI STRUCTURATE :

Un program este privit ca o multime ierarhica de blocuri si proceduri;

B: PARADIGMA PROGRAMARII ORIENTATE SPRE OBIECT: Un program este constituit dintr-o colectie de obiecte care interactioneaza;

C: PARADIGMA PROGRAMARII CONCURENTE SI DISTRIBUITE:

Executia unui program este constituita din actiuni multiple posibil a fi executate in paralel pe una sau mai multe masini;

D: PARADIGMA PROGRAMARII FUNCTIONALE: Un program este descris pe baza unor functii de tip matematic (fara efecte secundare), utilizate de obicei recursiv;

E: PARADIGMA PROGRAMARII LOGICE: Un program este descris printr-un set de relatii intre obiecte precum si de restrictii ce definesc cadrul in care functioneaza acele obiecte. Executia inseamna activarea unui proces deductiv.

F: PARADIGMA PROGRAMARII LA NIVELUL BAZELOR DE DATE:

Actiunile programului sunt dictate de cerintele unei gestiuni corecte si consistente a bazelor de date asupra carora actioneaza programul.

Caracteristici ale limbajului C

- limbaj eficient, portabil, permisiv**, poate fi **dificil de înțeles**
- limbaj eficient**
 - viteză mare de execuție a programelor, destinat și aplicațiilor implementate în limbaj de asamblare
 - reutilizarea ulterioară a subprogramelor
- limbaj portabil**
 - limbaj independent de hardware
- limbaj permisiv**
 - impune puține constrângeri, dă credit programatorului
 - permite introducerea unor erori care sunt foarte greu de depistat
- limbaj dificil de înțeles**
 - un stil de programare adecvat este foarte important
 - obfuscated C code contest: www.ioccc.org

```
#include      <stdio.h>
#define TA      q/*XYXY*/
#define/*X      YXY*/CG r=
void p(int   n,int c){;
for(;n--;)  putchar(c)
#define Y(    z)d;d=c++\
%2<1?x=x*4 +z,c%8>5?\n
x=x?p(1,x), 0:x:0:0;d=
#define/*X      YX*/C Y(1)
#define/*X      YX*/G Y(2)
;int(*f)( void),d,c,
#define/*X      YX*/A Y(0)
#define/*XY*/AT int\
m(void/**/){d=
#define/*XYX*/T Y(3)
#define GC  d; return\
0;}int(*f) (void )=m;
x,q,r; int main(){if(
f)f();else {for(puts(
"#include"
"\40\"pro\
g.c\"\n\n  \101T"+0);
d!=d?x=(x=
getchar())
<0?0:x,8*8 :d,TA++c%8
,TA(1+7*q-
q*q)/3,r=c
*15-c*c-36 ,p(r<0?q+
4:r/6+!q+4 ,32),q||x;
c%16)q?p( 1,"ACGT"[x
/d&3]),p(q ,126),p(1,
"TGCA"[x/d &3]),d/=4,
p(001,10): puts(c%8?\n
"CG":"TA") ;puts("GC"
);}return 0; }/**/
```

main.c

```
1 #include      <stdio.h>
2#define TA      q/*XYXY*/
3#define/*X      YXY*/CG r=
4 void p(int   n,int c){;
5 for(;n--;)  putchar(c)
6#define Y(    z)d;d=c++\
7%2<1?x=x*4 +z,c%8>5?\n
8x=x?p(1,x), 0:x:0:0;d=
9#define/*X      YX*/C Y(1)
10#define/*X      YX*/G Y(2)
11;int(*f)( void),d,c,
12#define/*X      YX*/A Y(0)
13#define/*XY*/AT int\
14m(void/**/){d=
15#define/*XYX*/T Y(3)
16#define GC  d; return\
170;}int(*f) (void )=m;
18x,q,r; int main(){if(
19f)f();else {for(puts(
20"#include"
21"\40\"pro\
22g.c\"\n\n  \101T"+0);
23d!=d?x=(x=
24getchar())
25<0?0:x,8*8 :d,TA++c%8
26,TA(1+7*q-
27q*q)/3,r=c
28*15-c*c-36 ,p(r<0?q+
294:r/6+!q+4 ,32),q||x;
30c%16)q?p( 1,"ACGT"[x
31/d&3]),p(q ,126),p(1,
32"TGCA"[x/d &3]),d/=4,
p(001,10): puts(c%8?\n
"CG":"TA") ;puts("GC"
);}return 0; }/**/
```

"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live."

~ John Woods

```
@statmethod
def calc_percent(byte_counter, data_len):
    if data_len is None:
        return '---.%'
    return '%G' % ('%.1f%%' % (float(byte_counter) / float(data_len) * 100.0))
```