



PROGRAMARE PROCEDURALĂ

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

Secția Informatică, anul I,

2016-2017

Cursul 1

Cuprinsul cursului de azi

1. Bun venit la Facultatea de Matematică și Informatică!
2. Utilitatea cursului de Programare Procedurală
3. Prezentarea cursului de Programare Procedurală
4. 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/2015/prezentare/
Prezentare_FMI_2015.pdf](http://fmi.unibuc.ro/ro/pdf/2015/prezentare/Prezentare_FMI_2015.pdf)
- Ghidul bobocului: <http://boboc.as-mi.ro/>

Bun venit!

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
- examen de licență – probă scrisă
- susținere lucrare licență – probă orală
- timpul trece repede: o săptămână $\approx 1\%$
- **regulă actuală: fără (prea multe) restanțe din anul 2 în anul 3**

Regulamente UB & FMI

Lucruri bine de știut de studenți:

- 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 așafoarte. În
continu foarte scură de o altă
și diferit, cu o altă cenușă).
în incidente evenimentul deține incident
major.

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

Blera

Utilitatea cursului de PP

- PP = paradigma de programare bazată pe conceptul de apel de procedură/funcție/rutină/subrutină. Un program este privit ca o mulțime 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.

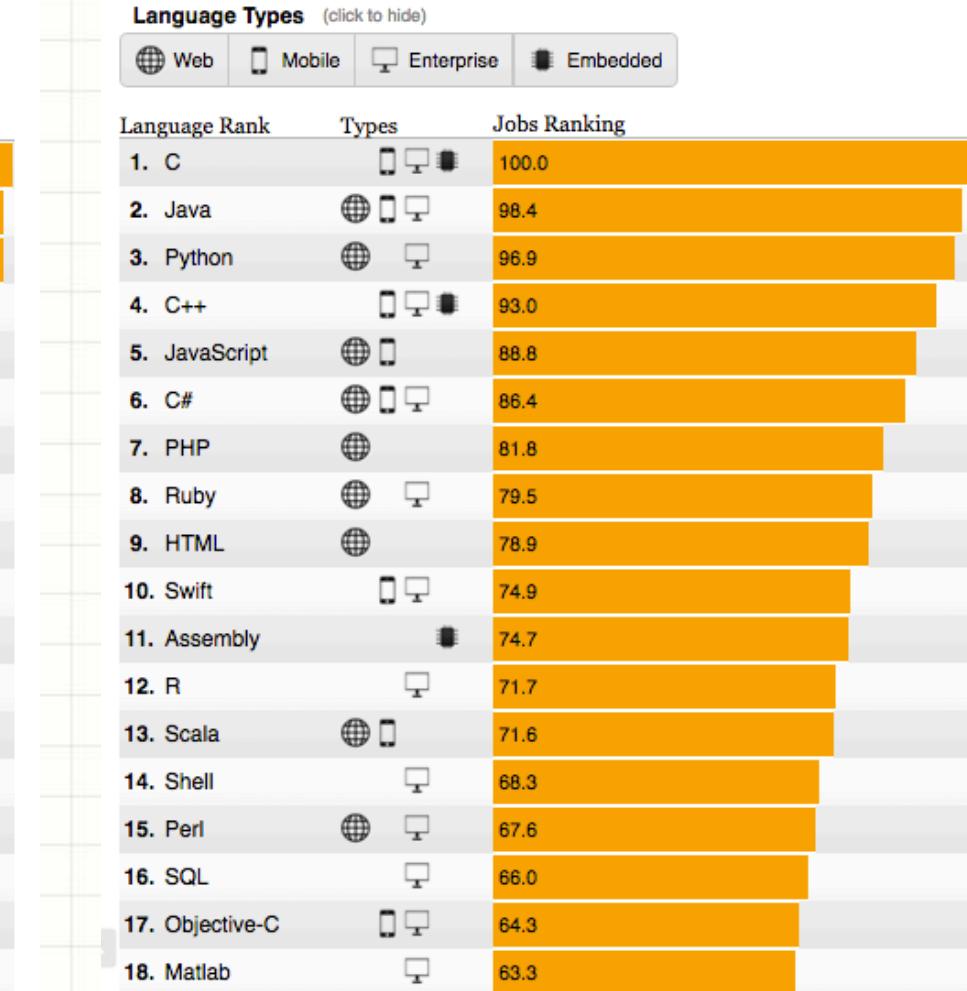
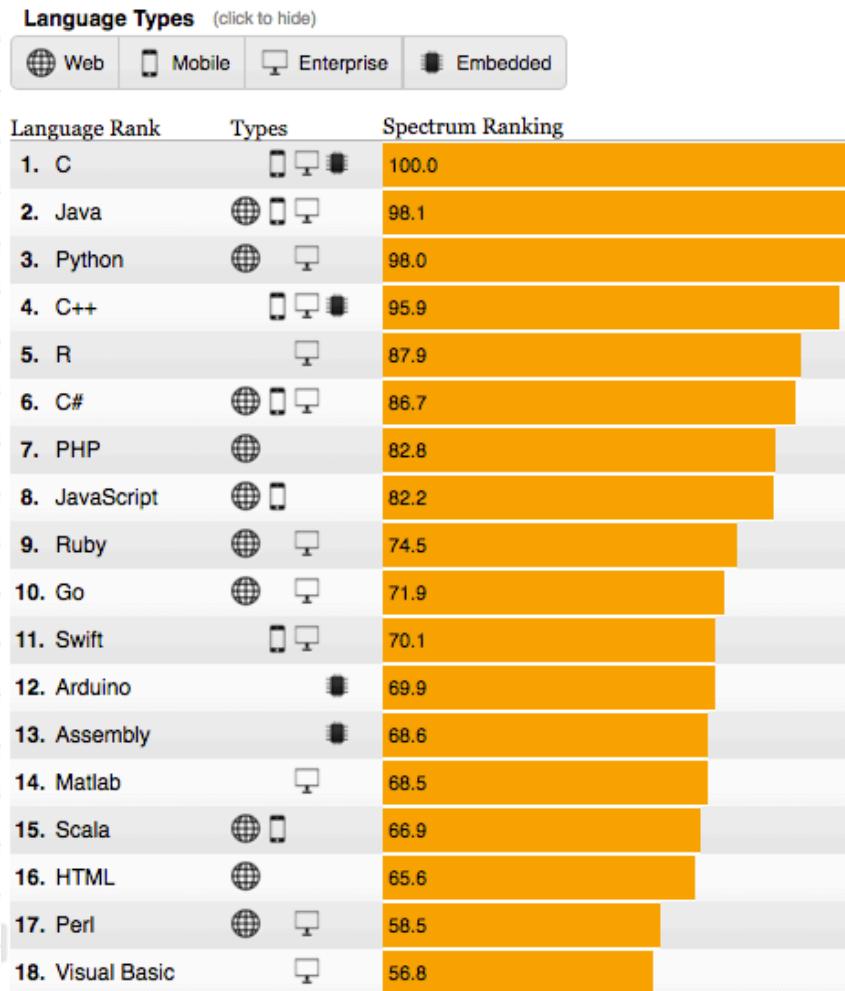
De ce C? <http://www.tiobe.com/tiobe-index/>

Sep 2016	Sep 2015	Change	Programming Language	Ratings	Change
1	1		Java	18.236%	-1.33%
2	2		C	10.955%	-4.67%
3	3		C++	6.657%	-0.13%
4	4		C#	5.493%	+0.58%
5	5		Python	4.302%	+0.64%
6	7	▲	JavaScript	2.929%	+0.59%
7	6	▼	PHP	2.847%	+0.32%
8	11	▲	Assembly language	2.417%	+0.61%
9	8	▼	Visual Basic .NET	2.343%	+0.28%
10	9	▼	Perl	2.333%	+0.43%
11	13	▲	Delphi/Object Pascal	2.169%	+0.42%
12	12		Ruby	1.965%	+0.18%
13	16	▲	Swift	1.930%	+0.74%
14	10	▼	Objective-C	1.849%	+0.03%
15	17	▲	MATLAB	1.826%	+0.65%
16	34	▲	Groovy	1.818%	+1.31%
17	14	▼	Visual Basic	1.761%	+0.23%
18	19	▲	R	1.684%	+0.64%
19	44	▲	Go	1.625%	+1.37%
20	18	▼	PL/SQL	1.443%	+0.36%

De ce C?

<http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016>

"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."



Prezentarea cursului de PP

- Structura primului semestru
- Orar
- Materiale
- Obiectivele cursului
- Programa cursului
- Bibliografie și resurse online
- Regulament de notare si evaluare
- Notele de anul trecut (secția Matematică)

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	03.10.2016 - 21.12.2016	Activitate didactică
		22.12.2016 – 04.01.2017	Vacanță de iarnă
		05.01.2017 – 21.01.2017	Activitate didactică
		22.01.2017 – 12.02.2017	Sesiune de examene
		13.02.2017 – 19.02.2017	Vacanță intersemestrială
		01.02.2017 - 15.02.2017	Sesiune de licență, disertație

- 13 cursuri de PP (1 decembrie fara curs): 10 cursuri în 2016 + 3 cursuri în 2017 (**cursul de pe 5 ianuarie se face!!!**)

Orar

Alexe Bogdan

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

	8	9	10	11	12	13	14	15	16	17	18	19
	8:00 - 8:50	9:00 - 9:50	10:00 - 10:50	11:00 - 11:50	12:00 - 12:50	13:00 - 13:50	14:00 - 14:50	15:00 - 15:50	16:00 - 16:50	17:00 - 17:50	18:00 - 18:50	19:00 - 19:50
Jo	ProgProced 135 3	ProgProced 131/132/133/134/135 3					ProgProced 131 3	ProgProced 134 9				
	ProgProced 133 3		0(Haret)				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

The screenshot shows the Moodle homepage for the Faculty of Mathematics and Informatics. The URL in the address bar is moodle.fmi.unibuc.ro. The page title is "Facultatea de Matematica si Informatica".

The left sidebar contains:

- A box for "Platforma educationala a Facultatii de Matematica si Informatica - Universitatea din Bucuresti".
- "Main menu" with "Site news" selected.
- "Navigation" with "Home" and "Courses". Under "Courses", there is a section "Zi" which is expanded, showing "Departament Informatica", "Adam Mircea", "Alexe Bogdan", "Co&AplInVedArtif", "M. Dr.", and "ProgProced1". The "ProgProced1" link is highlighted with a red oval.

The main content area is titled "Site news" and contains the following text:

Autentificare
by Administrator Moodle - Saturday, 22 September 2012, 08:12 PM

Pentru a vă recupera user-ul și parola, contactați-vă șeful de grupă, el trebuie să aibă lista și să vă informeze.

Vă rugăm, la prima logare să vă schimbați emailul din profilul dumneavoastră cu unul existent pentru a ușura recuperările de parola.

Dacă întâmpinați probleme trimiteți un email la moodle@fmi.unibuc.ro.

Mulțumim!

Obiectivele cursului

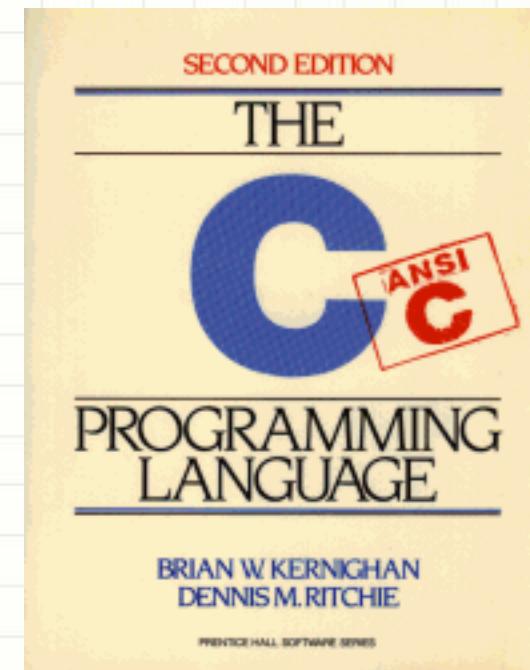
1. Formarea deprinderilor de **programare structurată** în limbaje de programare clasice și moderne (descompunerea unei probleme complexe în subprobleme relativ simple și independente);
2. Însușirea caracteristicilor **limbajului C**: alocarea memoriei, lucrul cu pointerii, lucrul cu fișierele, programarea 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. Deprinderea tehnicii de **testare și de verificare** a corectitudinii programelor;
4. 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.
 - Instrucțiuni de control
 - Directive de preprocesare. Macrodefiniții.
 - Funcții de citire/scriere.
 - Etapele realizării unui program C.
- Tipuri deriveate de date**
 - Tablouri. Siruri de caractere.
 - Structuri, uniuni, câmpuri de biți, enumerări.
 - Pointeri.
- Funcții (1)**
 - Declarare și definire. Apel. Metode de transmisie a parametrilor.
 - Pointeri la funcții.
- Tablouri și pointeri**
 - Legătura dintre tablouri și pointeri
 - Aritmetică a pointerelor
 - Alocarea dinamică a memoriei
 - Clase de memorare
- Siruri de caractere**
 - Funcții specifice de manipulare.
- Fișiere text și 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

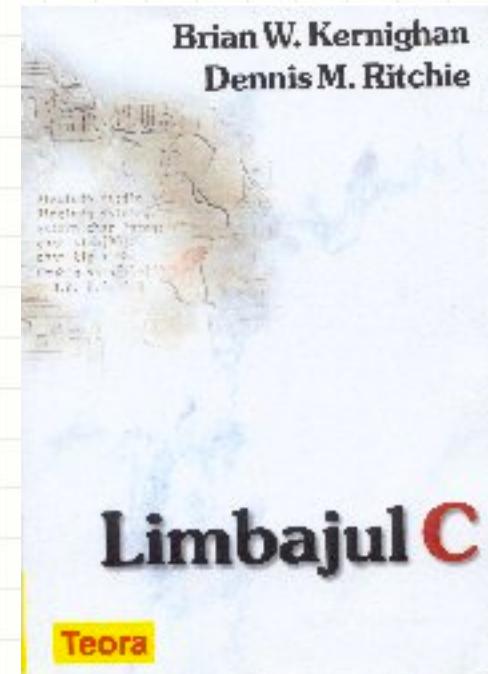
1. Kernighan & Ritchie: The C programming language
<http://zanasi.chem.unisa.it/download/C.pdf>



Bibliografie

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

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

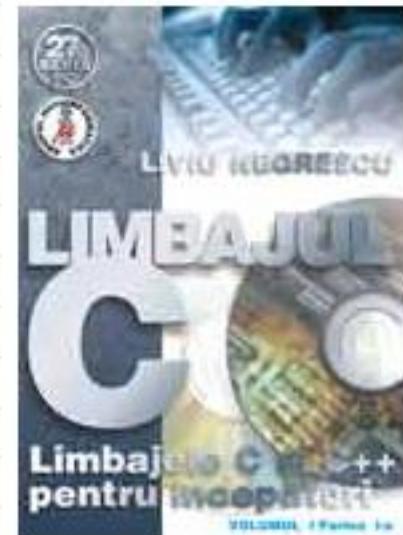


Bibliografie

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



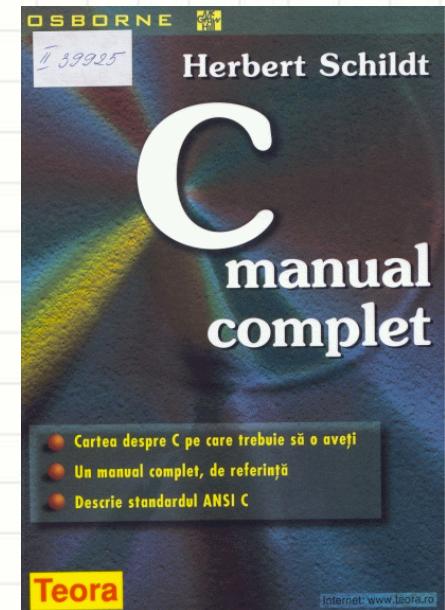
Bibliografie

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

InfO'Clock

- curs facultativ ținut la FMI de 1-2 ori/săptămână și adresat studenților pasionați de programare, algoritmică, structuri de date, tehnici de programare care vor să participe la concursuri
- nivel foarte ridicat, se rezolvă probleme de concursuri
- participarea la acest curs: teme + concursuri online sau organizate la FMI

Regulament de evaluare și notare

$$Nota = \min (10, 1 + Curs + Laborator + Seminar)$$
$$4p \qquad \qquad \qquad 5p \qquad \qquad \qquad 1p$$

Curs (4 puncte): examen scris se dă în sesiune o singură dată cu toți studenții care au intrat în examen.

Laborator (5 puncte): teme (3 puncte) + test (2 puncte) în săptămâna 14 cu toată seria.

Seminar (1 punct): 1 punct pentru activitate, posibil o lucrare.

Nu intrați în examen (=restanță) dacă:

- nu luați peste 5 (1 punct) la testul final din săptămâna 14;
- nu acumulați peste 2.5 puncte la laborator (din teme + test).

Aveți restanță dacă:

- nu intrați în examen;
- nu luați peste nota 5 (2 puncte) la curs.

Restanță – iunie și septembrie

$$Nota = \min (10, 1 + Curs + Laborator + Seminar)$$
$$\qquad\qquad\qquad 4p \qquad\qquad\qquad 5p \qquad\qquad\qquad 1p$$

Curs (4 puncte): examen scris.

Laborator (5 puncte): dacă ați luat laboratorul (minim 2.5 puncte la teme și test (minim 1 punct)) vi se păstrează nota, altfel dați un test (5 puncte)

Seminar (1 punct): vi se păstrează nota din timpul semestrului

Restanță – anul universitar 2017-2018

$$Nota = \min (10, 1 + Curs + Laborator + Seminar)$$

Curs *Laborator* *Seminar*

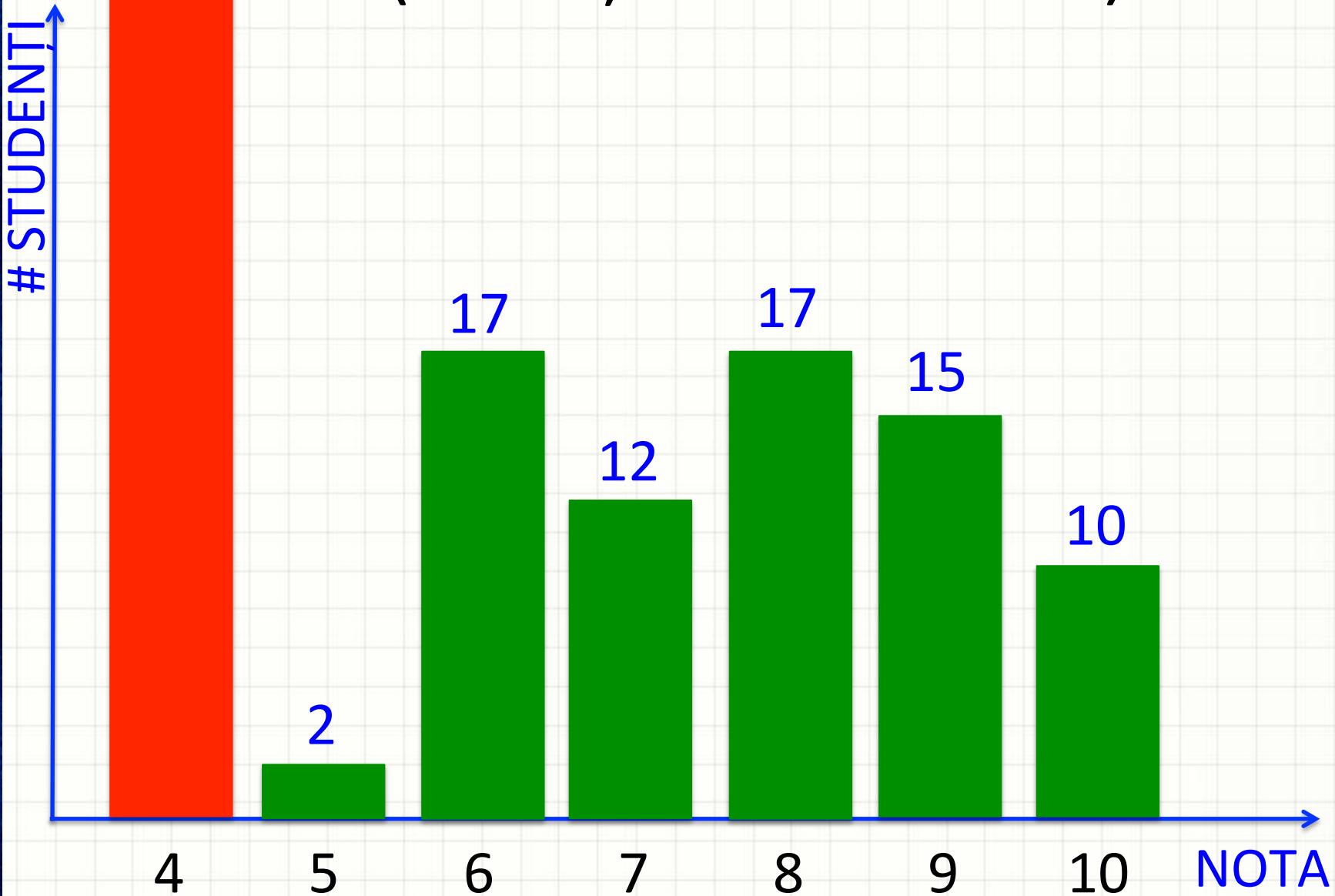
4p 5p 1p

Curs (4 puncte): examen scris.

Laborator (5 puncte): refaceti laboratorul

Seminar (1 punct): refaceti seminarul

Notele de anul trecut - ianuarie (la secția Matematică)



Compilatoare vs IDE

Compilator = program care transformă codul sursă al unui program scris într-un limbaj de programare în cod mașină.
Exemple: gcc, minGW, clang, etc

IDE = integrated development environment = mediu de dezvoltare care pune la dispoziția programatorului un editor pentru codul sursă, depanator, compilator
Exemple: Code::Blocks, Visual Studio, Eclipse, Dev-C++, etc

Code::Blocks

www.codeblocks.org/downloads



Code::Blocks

Code::Blocks - The IDE with all the features you need, having a consistent look, feel and operation across platforms.

[Home](#) [Features](#) [Downloads](#) [Forums](#) [Wiki](#)

Main

- Home
- Features
- Screenshots
- Downloads
 - Binaries
 - Source
 - SVN
- Plugins
- User manual
- Licensing
- Donations

Downloads

There are different ways to download and install Code::Blocks on your computer:

- [Download the binary release](#)

This is the easy way for installing Code::Blocks. Download the setup file, run it on your computer and Code::Blocks will be installed, ready for you to work with it. Can't get any easier than that!

- [Download a nightly build](#): There are also more recent so-called *nightly builds* available in the [forums](#) or (for Debian and Fedora users) in [Jens' Debian repository](#) and [Jens' Fedora repository](#). Other distributions usually follow provided by the community (Big "Thank you" for that!) Please note that we consider nightly builds to be *stable*, usually, unless stated otherwise.

- [Download the source code](#)

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 ...

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

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



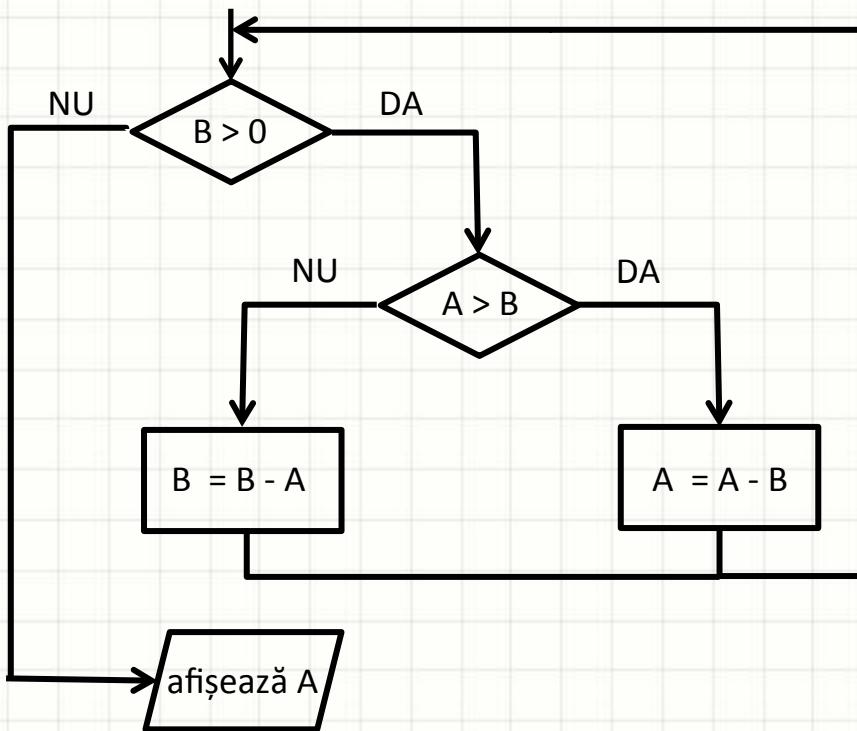
Direcția fluxului



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

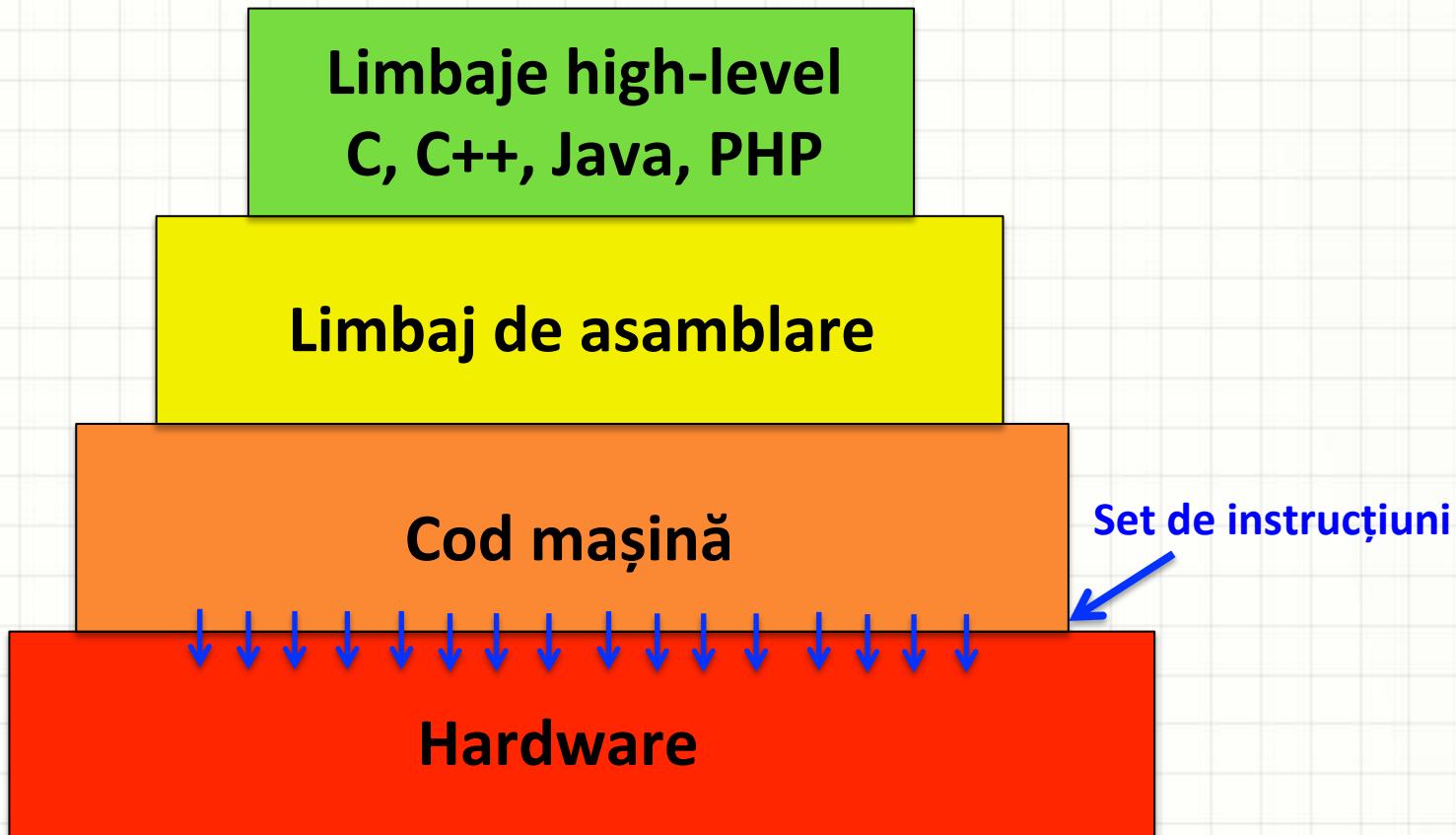
Limbaje de programare

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

Instructiuni în limbaj mașină

Limbaj de asamblare

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

```
swap:  
    mul $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 in
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 declare 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  
000000000000110000000110000001000001  
10001100011000100000000000000000000000  
10001100111100100000000000000000000000100  
10101100111100100000000000000000000000000  
101011000110001000000000000000000000000100  
0000001111110000000000000000000000000000000
```

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.

Paradigma programării procedurale

- execuție secvențială
- variabile reprezentate ca poziții în memorie și modificate prin atribuiri
- unitatea de program de bază: procedura = funcția = rutină = subrutină = subprogram
- C, Pascal, Fortran, Basic, Algol

Cursul 1:

1. Algoritmi

2. Limbaje de programare

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

Limbajul C

- ❑ popular, rapid și independent de platformă
- ❑ este un limbaj utilizat cel mai adesea pentru scrierea programelor eficiente și portabile: sisteme de operare, aplicații embedded, compilatoare, interpretoare, etc.
- ❑ limbajul C a fost dezvoltat la începutul anilor 1970 în cadrul Bell Laboratories de către Dennis Ritchie
 - ❑ strâns legat de sistemele de operare UNIX
- ❑ stă la baza pentru majoritatea limbajelor "moderne": C++, Java, C#, Javascript, Objective-C, etc.

Limbajul C

- ❑ trei **standarde oficiale active ale limbajului**
 - ❑ **C89** (C90) – aprobat în 1989 de ANSI (American National Standards Institute) și în 1990 de către ISO (International Organization for Standardization)
 - ❑ C89 a eliminat multe din incertitudinile legate de sintaxa și gramatica limbajului.
 - ❑ cele mai multe compilatoare de C sunt compatibile cu acest standard (ANSI C)
 - ❑ **C99** – standard aprobat în 1999, care include corecturile aduse C89 dar și o serie de caracteristici proprii care în unele compilatoare apăreau ca extensii ale C89 până atunci
 - ❑ compilatoarele oferă suport limitat și în multe cazuri incomplet pentru acest standard
 - ❑ **C11** – standard aprobat în 2011 și care rezolvă erorile apărute în standardul C99 și introduce noi elemente, însă suportul pentru C11 este și mai limitat decât suportul pentru C99, majoritatea compilatoarelor nu s-au adaptat încă la acest standard

Caracteristici ale limbajului C

- ❑ limbaj procedural, structurat, compilat, de nivel de mijloc, scurt
- ❑ limbaj procedural, structurat
 - ❑ instrucțiuni specificate sub forma unor comenzi grupate într-o ierarhie de subprograme (denumite funcții) și care pot forma module
- ❑ limbaj compilat
 - ❑ compilatorul transformă instrucțiunile limbajului C în limbaj mașină
- ❑ limbaj de nivel de mijloc
 - ❑ permite accesul la date și comenzi aflate aproape de nivelul fizic folosind o sintaxă specifică limbajelor de nivel înalt
- ❑ limbaj scurt
 - ❑ număr redus de cuvinte cheie
 - ❑ multe funcționalități nu sunt incluse în limbajul de bază ci necesită includerea unor biblioteci standard

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))
```