



# PROGRAMAREA CALCULATOARELOR

Andrei Pătrașcu

[andrei.patrascu@fmi.unibuc.ro](mailto:andrei.patrascu@fmi.unibuc.ro)

Cursul 1

# Cuprinsul cursului de azi

1. Prezentarea cursului de Programarea Calculatoarelor
2. Primul curs

# Utilitatea cursului de PC

- 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.

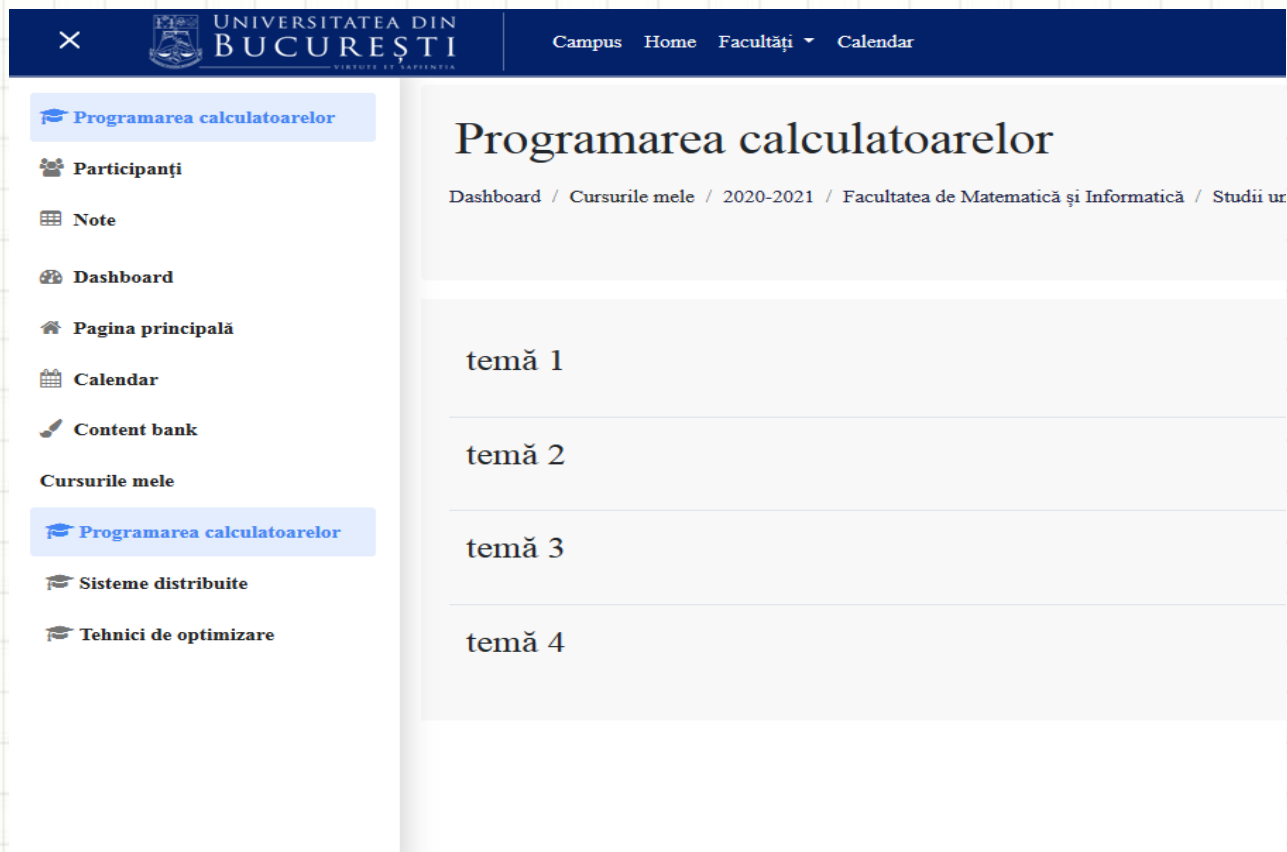
# De ce C?

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

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.

# Materiale

- MSTeams si/sau moodle.unibuc.ro



The screenshot displays the Moodle interface for the University of Bucharest. The top navigation bar is dark blue with the university's logo and name, 'UNIVERSITATEA DIN BUCUREȘTI', and a menu with 'Campus', 'Home', 'Facultăți', and 'Calendar'. The left sidebar contains a list of course-related links: 'Programarea calculatoarelor' (highlighted), 'Participanți', 'Note', 'Dashboard', 'Pagina principală', 'Calendar', 'Content bank', 'Cursurile mele', 'Sisteme distribuite', and 'Tehnici de optimizare'. The main content area shows the course title 'Programarea calculatoarelor' and a breadcrumb trail: 'Dashboard / Cursurile mele / 2020-2021 / Facultatea de Matematică și Informatică / Studii ur'. Below this, there is a list of four topics: 'temă 1', 'temă 2', 'temă 3', and 'temă 4', each in a light blue box.



# 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, șiruri 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 paramerilor. Pointeri la funcții.

## □ Tablouri și pointeri

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

## □ Șiruri de caractere

- Funcții specifice de manipulare.

## □ Fișiere binare

- Funcții specifice de manipulare.

## □ Structuri de date complexe și autoreferite

- Definiție și utilizare

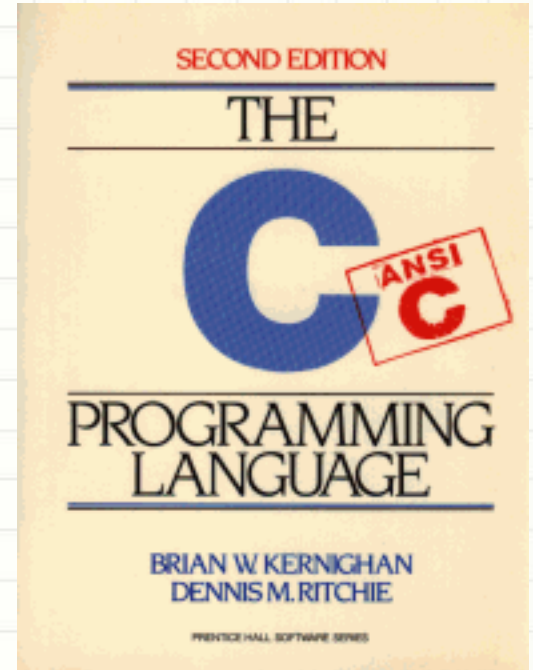
## □ Funcții (2)

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

# Bibliografie

1. Kernighan & Ritchie: The C programming language

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





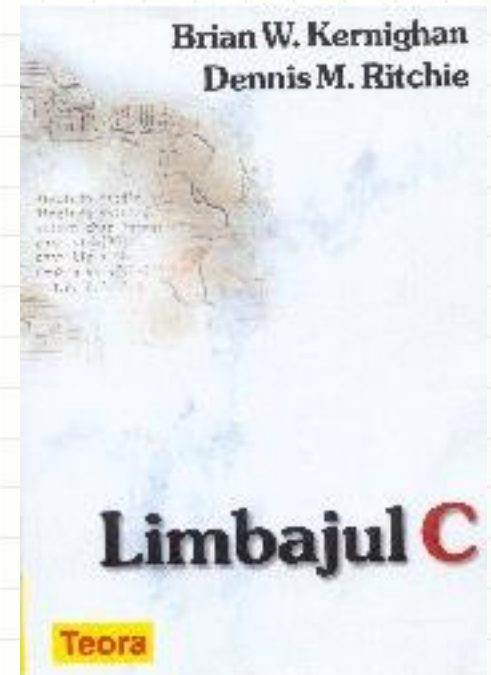
# Bibliografie

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

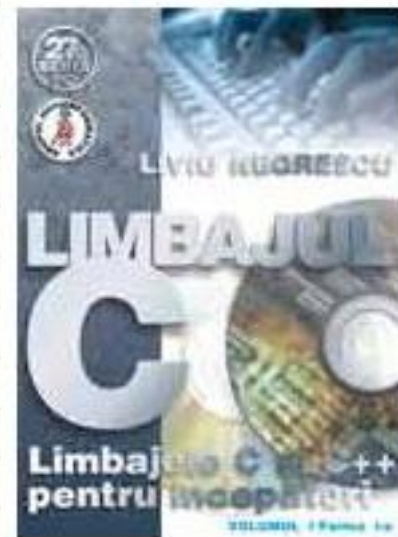
<http://zanasi.chem.unisa.it/download/C.pdf>

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

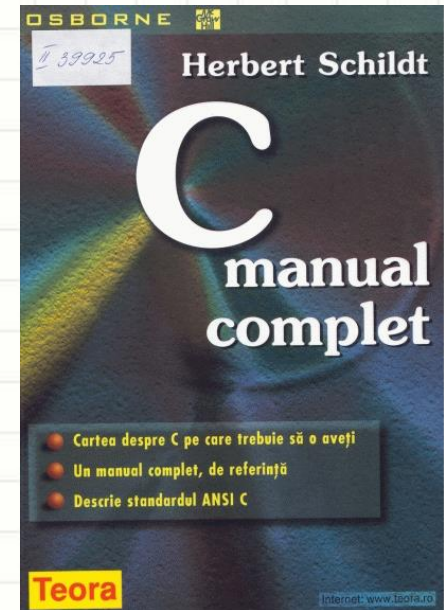
Editura Teora, 2003

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>

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)

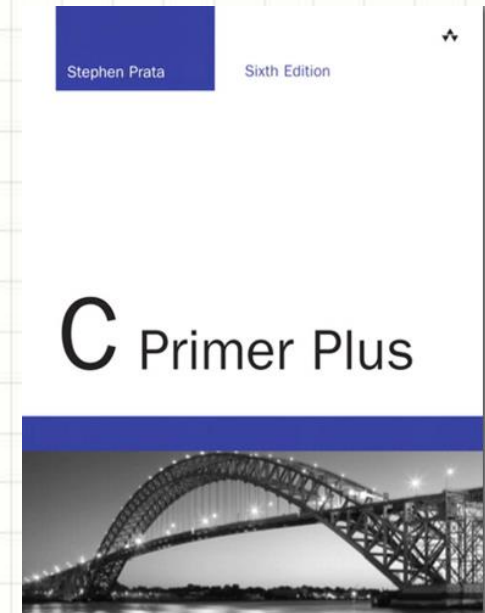
Editura Albastra, 2001

4. Herbert Schildt: C, manual complet.

Editura Teora, 2000?

5. Stephan Prata: C primer plus, 6<sup>th</sup> Edition

[https://vk.com/doc190970339\\_430409589?hash=2d2b4245bd65b25e27&dl=cd4e96f98aeddd5c1e](https://vk.com/doc190970339_430409589?hash=2d2b4245bd65b25e27&dl=cd4e96f98aeddd5c1e)



# Regulament de evaluare și notare

$$Nota = \min(10, \underset{6p}{Curs} + \underset{4p}{Laborator} + \underset{1p}{Seminar})$$

Planul evaluării nu este gata încă!

Detalii despre modul de evaluare în cursul 2!



# Mulțumiri pentru materiale/slide-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. Prezentarea cursului de Programare Procedurală
2. 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, șiruri 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 parametrelor. Pointeri la funcții.

## ☐ Tablouri și pointeri

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

## ☐ Șiruri 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.

# 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 repetate, 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



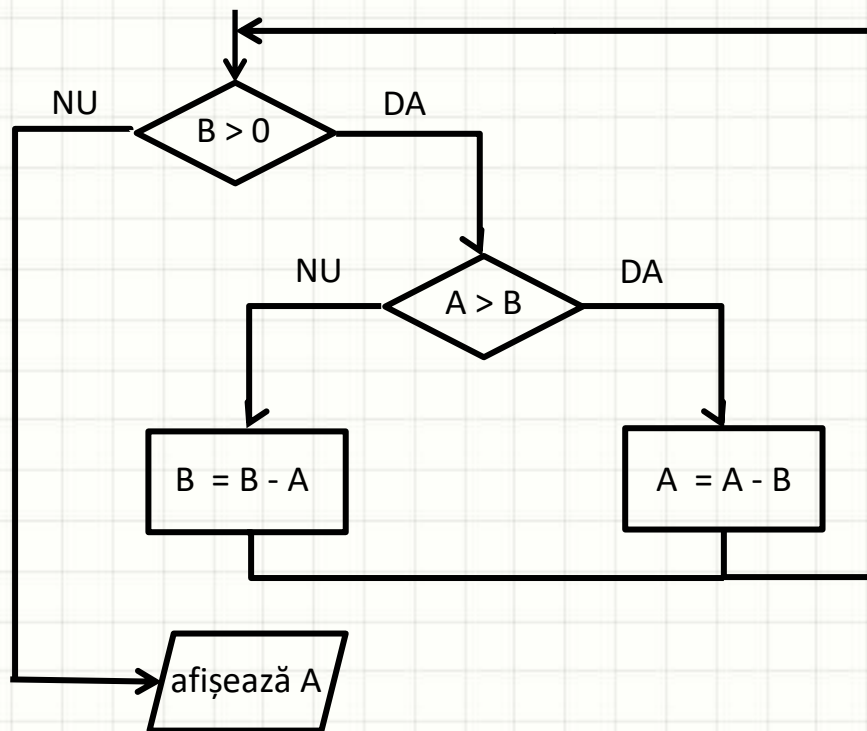
Direcția fluxului



Operația de intrare/ieșire

# Schemă logică

- alăturare de simboluri vizuale care desemnează fluxul logic al pașilor



*Algoritmul lui Euclid  
prin scăderi repetate*

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:

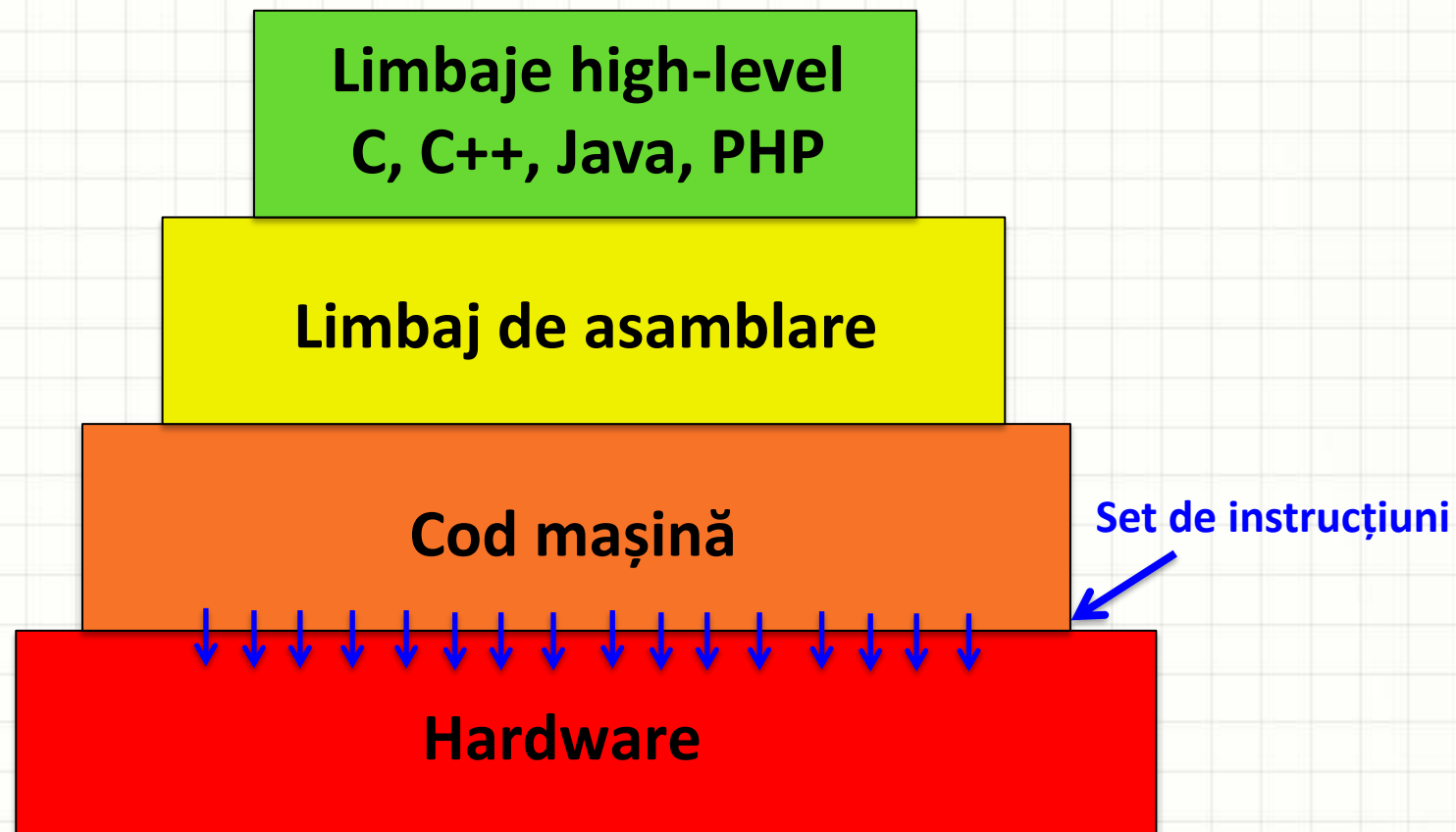
1. Analiza problemei
2. Găsirea soluției [optime]
3. Elaborarea algoritmului
4. Implementarea algoritmului într-un limbaj de programare
5. Verificarea corectitudinii algoritmului propus
6. Analiza complexității

# Limbaaj de programare

- ❑ limbaj artificial cu sintaxă și semantică bine definite
- ❑ pune la dispoziția programatorilor construcții sintactice prin care sunt specificate sucesiunea 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);
- ❑ șabloane de numere binare (reprezintă modul binar de codificare a instrucțiunilor și datelor în memorie )
- ❑ depinde de arhitectura sistemului

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
100011001111001000000000000000100
10101100111100100000000000000000
101011000110001000000000000000100
00000011111000000000000000001000
```

Instrucțiuni 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:
    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



# Limbaje high-level (de nivel înalt)

- ❑ cuprind mecanisme de exprimare apropiate 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 și definire.

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

Instrucțiuni 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

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
0000001111100000000000000001000
```

Instrucțiuni 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

# 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](http://www.ioccc.org)

# Cuvinte cheie

C89 = ANSI C : 32 de cuvinte cheie

|          |        |          |          |
|----------|--------|----------|----------|
| auto     | double | int      | struct   |
| break    | else   | long     | switch   |
| case     | enum   | register | typedef  |
| char     | extern | return   | union    |
| const    | float  | short    | unsigned |
| continue | for    | signed   | void     |
| default  | goto   | sizeof   | volatile |
| do       | if     | static   | while    |

C99: ANSI C + alte 5 cuvinte cheie

`_Bool` `_Complex` `_Imaginary` `inline` `restrict`

# Structura generală a unui program C

- ❑ modul principal (funcția main)
- ❑ zero, unul sau mai multe module (funcții/proceduri) care comunică între ele și/sau cu modulul principal prin intermediul parametrilor și/sau a unor variabile globale
- ❑ unitatea de program cea mai mică și care conține cod este funcția/procedura și conține:
  - partea de declarații/definiții;
  - partea imperativă (comenzile care se vor executa);

# Primul program C

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Primul program scris in C\n");
6      return 0;
7  }
```



# Primul program C explicat

```
1 #include <stdio.h>
```

← Directivă de preprocesare pentru  
includerea bibliotecii standard de i/o

Antetul funcției principale

```
3 int main()
```

```
4 {  
5     printf("Primul program scris in C\n");  
6     return 0;  
7 }
```

← Funcția principală

Corpul funcției

## Observații:

- ☐ **main** nu este cuvânt cheie în limbajul C, îl utilizăm pentru numirea funcției principale;
- ☐ **printf** nu este cuvânt cheie, este funcție de bibliotecă (print (afișare) +f (format));
- ☐ C este case sensitive, se face diferență între litere mici și mari;
- ☐ toate cuvintele cheie se scriu cu litere mici;
- ☐ instrucțiunile se termină cu **caracterul ;** (punct și virgulă);
- ☐ mai multe instrucțiuni pot fi scrise pe aceeași linie;
- ☐ **spațiile** ajută la organizarea codului.

# Structura unui program C simplu

*directive de preprocesare*

```
int main()  
{  
    instrucțiuni  
}
```

```
1  #include <stdio.h>  
2  
3  int main()  
4  {  
5      printf("Primul program scris in C\n");  
6      return 0;  
7  }
```

## □ Directive de preprocesare

- directive de definiție: #define N 10
- directive de includere a bibliotecilor: #include <stdio.h>
- directive de compilare condiționată: #if, #ifdef, ...
- alte directive (vorbim în cursurile următoare)

## □ Funcții

- grupări de instrucțiuni sub un nume;
- returnează o valoare sau se rezumă la efectul produs;
- funcții scrise de programator vs. funcții furnizate de biblioteci;
- programul poate conține mai multe funcții;
  - **main** este obligatoriu;
- antetul și corpul funcției.

# Structura unui program C simplu

*directive de preprocesare*

```
int main()  
{  
    instrucțiuni  
}
```

```
1 #include <stdio.h>  
2  
3 int main()  
4 {  
5     printf("Primul program scris in C\n");  
6     return 0;  
7 }
```

## ❑ Instrucțiuni

- ❑ formează corpul funcțiilor
  - ❑ exprimate sub formă de comenzi
- ❑ 5 tipuri de instrucțiuni:
  - ❑ instrucțiunea declarație;
  - ❑ instrucțiunea atribuire;
  - ❑ instrucțiunea apel de funcție;
  - ❑ instrucțiuni de control;
  - ❑ instrucțiunea vidă;
- ❑ toate instrucțiunile (cu excepția celor compuse) se termină cu caracterul ";"
  - ❑ caracterul ; nu are doar rol de separator de instrucțiuni ci instrucțiunile încorporează caracterul ; ca ultim caracter
  - ❑ omiterea caracterului ; reprezintă eroare de sintaxă

# Structura unui program C complex

*comentarii*

*directive de preprocesare*

*declarații și definiții globale*

```
int main()
```

```
{
```

*declarații și definiții locale*

*instrucțiuni*

```
}
```

# Structura unui program C complex

## Comentariile

- formă de documentare a codului sursă, sunt ignorate de compilator
- 2 tipuri de comentariu:
  - începe cu `/*` și se termină cu `*/`: se pot extinde pe mai multe linii, nu se pot imbrica, sunt utile pentru inserarea unor explicații mai lungi
  - începe cu `//` și se termină la sfârșitul liniei: utile pentru comentariile inserate pe marginea codului (apare în C99, nu este în C89)



# Cod laborator

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main()
6  {
7
8  char a,b;
9
10 scanf ("%d %d",&a,&b);
11 printf ("a = %d\n",a);
12 printf ("b = %d\n",b);
13
14
15
16 return 0;
17 }
18
```

# Cod laborator

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main()
6  {
7
8  char a,b;
9
10 scanf ("%d %d",&a,&b);
11 printf("a = %d\n"
12 printf("b = %d\n"
13
14
15
16 return 0;
17 }
18
```

"D:\Work\Cursuri\Programarea Calculatoarelor 2020\Laborator 2021\Lab1\lucru\Laborator1

```
23 55
a = 0
b = 55

Process returned 0 (0x0)   execution time : 6.883 s
Press any key to continue.
```

# Cod laborator

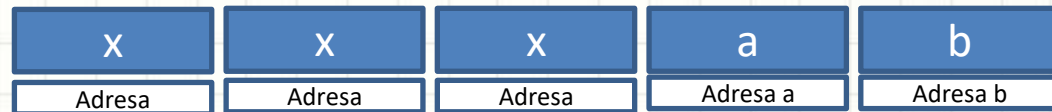
```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main()
6  {
7
8  char a,b;
9
10 scanf ("%d %d", &a, &b);
11 printf ("a = %d\n", a);
12 printf ("b = %d\n", b);
13
14
15
16 return 0;
17 }
18
```

Memorie **int: 4 octeti**

Instructiunea de citire:

**scanf("%d",&a)**

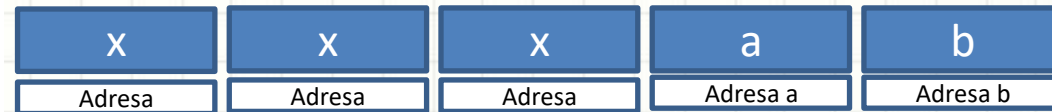
- scrie 4 octeti in memorie incepand cu adresa lui a
- modifica variabila **a**, dar si locatiile vecine



# Cod laborator

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main()
6  {
7
8  char a,b;
9
10 scanf ("%d %d",&a,&b);
11 printf("a = %d\n",a);
12 printf("b = %d\n",b);
13
14
15
16 return 0;
17 }
18
```

Memorie (stare initiala)



# Cod laborator

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main()
6  {
7
8  char a,b;
9
10 scanf ("%d %d",&a,&b);
11 printf("a = %d\n",a);
12 printf("b = %d\n",b);
13
14
15
16 return 0;
17 }
18
```

Memorie (stare initiala)



Memorie (dupa executia scanf)





# Cod laborator

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  int main()
6  {
7
8  char a,b;
9
10 scanf("%d %d",&a,&b);
11 printf("a = %d\n",a);
12 printf("b = %d\n",b);
13
14
15
16 return 0;
17 }
18
```

Memorie (stare initiala)



Memorie (dupa executia scanf)



Rezultat afisare:

