



# Fundamentele programării

Conf. Dr. Elena BĂUTU  
Prof. Dr. Dorin Mircea  
POPOVICI

# Organizare

- Curs: 2 ore
- Laborator: 2 ore
- Numar credite: 6
- Studiu individual: 94 ore/semestru
- Titulari disciplina: Conf. dr. Elena Băutu
- Alumni of Fac. De Informatică, Univ. Al.I.Cuza, Iași
  - PhD in Computer Science (Artificial Intelligence)
  - Contact Email: ebautu@univ-ovidius.ro
  - Url: <https://sites.google.com/site/elenabautu/>

# Evaluare

- Participare activă la activități didactice: 10%
- Teme de casă: 10%
- Teste la laborator: 30%
- Examen: 50%

Nota la examen se calculează ca medie ponderată.

Testele de laborator se dau în timpul semestrului și nu se reiau în nicio sesiune de examinare.

Modalitatea de calcul a notei este aceeași, indiferent de sesiunea de examinare la care participa studentul.

Examenul se considera promovat dacă media la examen este mai mare sau egală cu 5.

# Objective

- Insusirea celor mai importante concepte de programare
- Familiarizarea cu termeni din ingineria software (e.g. arhitectura, implementare, intretinere, mediu de dezvoltare)
- Familiarizarea cu instrumente software de baza
- Invatarea limbajului de programare C
- Folosirea limbajului C pentru implementarea, rulara, testarea si depanarea programelor
- Insusirea/imbunatatirea stilului de programare

# Curs

---

Concepte de bază.

---

Operatori. Instrucțiuni (if, while, do-while, for).

---

Tablouri de date.

---

Funcții. Recursivitate.

---

Pointeri

---

Lucrul cu fișiere.

---

Structuri.

# Laborator

---

Aplicații la noțiunile prezentate la curs.

---

Rezolvarea de probleme

---

Cunoașterea conceptelor fundamentale

---

Abilitati de programare

# Bibliografie

---

[1] Suportul de curs si cel de laborator, disponibile pe <https://moodle.univ-ovidius.ro>

---

[2] Limbajele C si C++ pentru incepatori, Liviu Negrescu, Editura Albastra, Cluj Napoca, 2000

---

[3] Brian Kernighan and Dennis Ritchie, [The C Programming Language Book](#), (1988)

---

[4] Joel Sommers, [The Book of C](#) (2022)

---

[5] Tutorial C, <https://www.tutorialspoint.com/cprogramming/>

---



# Cursul 1

Introducere în programarea  
calculatoarelor



# Sumar

## Structura unui program

### Instructiuni simple:

- Atribuire, citire/scriere

### Date si Operatii

- operatori aritmetici
- instructiunea if
- operatori logici

# Programarea calculatoarelor

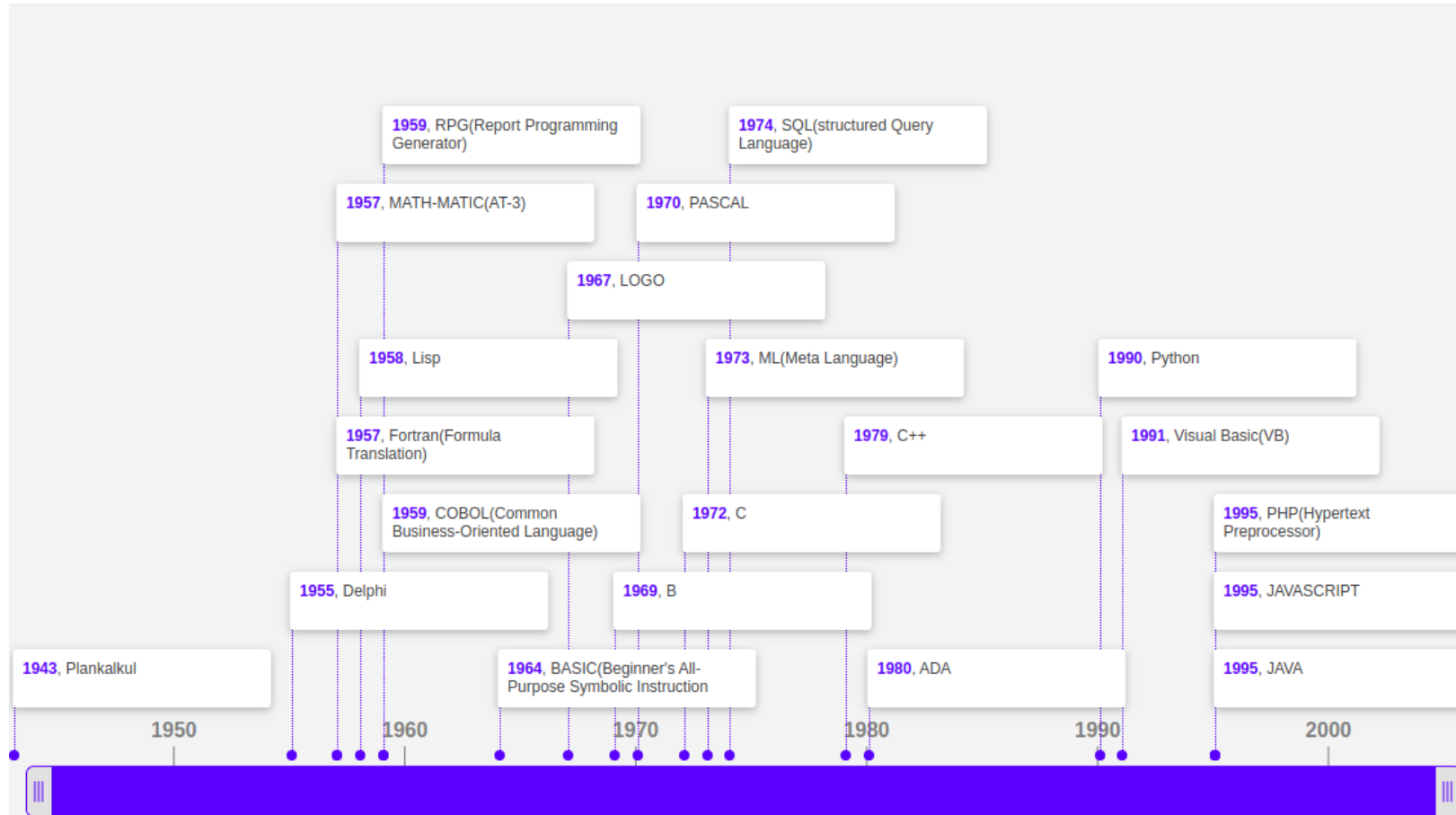
## Scrierea de programe software

- Analiza + specificarea
- Proiectarea
- Implementarea
- Testarea + repararea erorilor
- Documentarea
- Intreținerea

## Paradigme de programare

- Programare structurată
- **Programare imperativă (procedurală)**
- Programare orientată spre obiecte
- Programare declarativă
  - Programare funcțională
  - Programare logică
  - Programare bazată pe reguli

# Scurt istoric al evolutiei limbajelor de programare



Sursa: <https://www.timetoast.com/timelines/history-of-programming-languages-d13a48c2-eb47-4ddc-b64c-1d046361efba>

# Program

## Ce este un program?

- O insiruire de instructiuni ce implementeaza un algoritm de rezolvare a unei probleme

## Limbaj de programare

- Ansamblu de notatii si reguli pentru definirea sintaxei si semanticii unui program

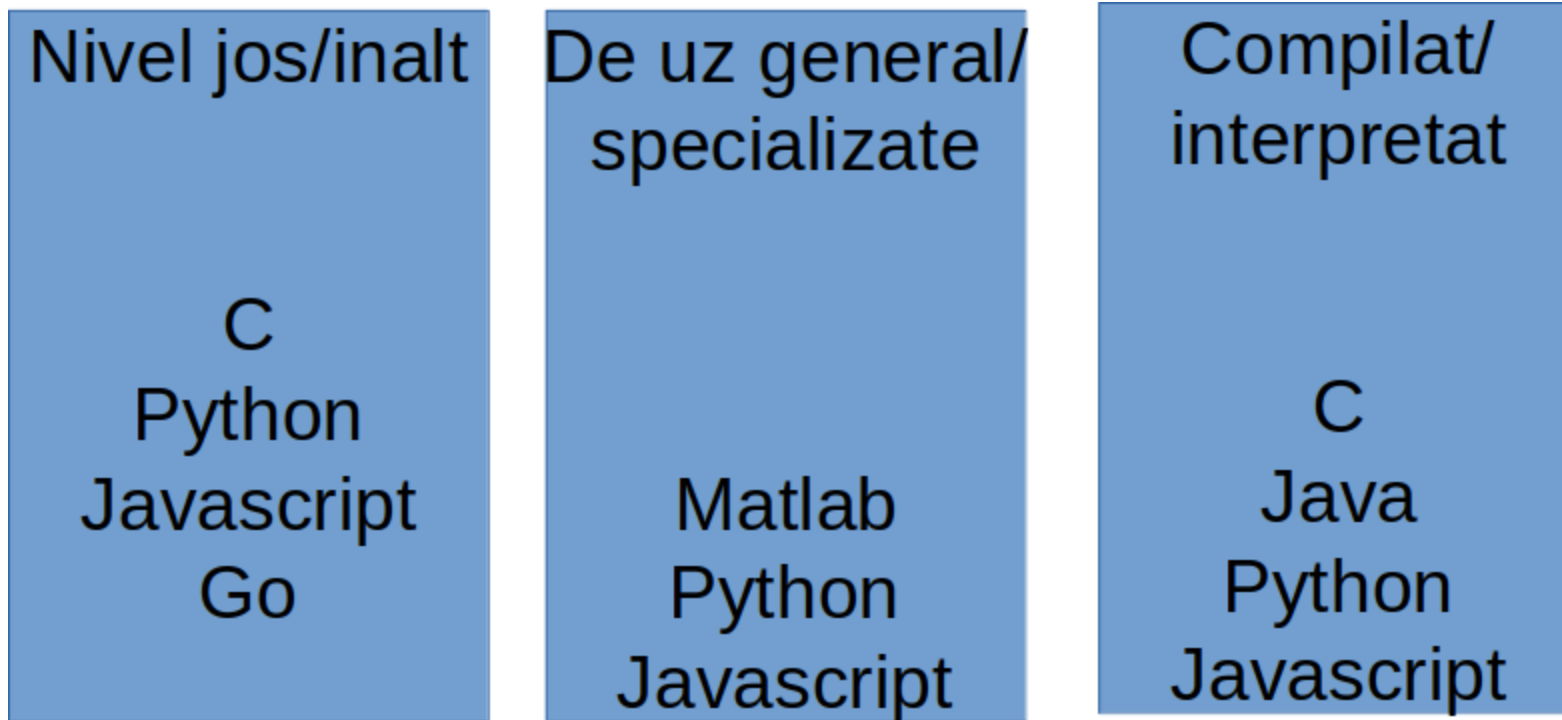
## Cum obtinem un program?

- Fisier sursa -> (cod obiect) -> cod masina

## *Eu* pot scrie un program?

- Perseverand, "DA"

# Limbaje de programare



# Limbajul C

- limbaj de programare
  - nespecializat (folosit în diverse domenii)
  - nivel mediu (ușor de învățat, relativ ușor de folosit)
  - foarte popular (datorită portabilității sale)
  - predecesorul multor limbaje (C++, PHP, Java, C#)
- C18 – varianta standard ISO (C99 – cel mai utilizat)
  - <https://www.iso.org/standard/74528.html>
  - folosește fișiere sursă cu extensia .c
  - .cpp pentru C++
- C++
  - Clase, moștenire, argumente implicite, inline, funcții virtuale, supraîncărcare, referințe, const

# Realizarea unui program C

- Identificarea/conceperea algoritmului
- Transcrierea algoritmului în limbajul C (folosind un mediu de programare)
  - E.g. Dev-cpp, Code Blocks, [https://www.onlinegdb.com/online\\_c\\_compiler](https://www.onlinegdb.com/online_c_compiler)
- Compilarea programului
  - fișierul sursă C se transformă în cod masina, dupa care poate fi executat
- Testarea programului
  - corectarea erorilor

# Primul program C

```
// primul meu program C
```

**A - directive, librarii**

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

**B - declaratii, ct, fctii, tipuri**

```
int main()
```

```
{
```

```
    printf("Bine ati venit!\n");
```

```
    return 0;
```

```
}
```

**C - functia main()**

**D - implementarea functiilor declarate in zona B**



# Primul program C

```
// primul meu program C++
#include <iostream.h>
using namespace std;



int main()
{
    cout << "Bine ati venit!" << endl;
    return 0;
}
```

**A - directive, librarii**


**B - declaratii, ct, fctii, tipuri**

**C - functia main()**

**D - implementarea functiilor declarate in zona B**



Alfabet.  
Vocabular.  
Unitati lexicale.



# Mulțimea caracterelor

- similar cu **alfabetul** unei limbi
  - fiecare caracter are un înțeles special pentru compilatorul C
- limbajul C folosește caracterele ASCII
  - litere (mici și mari din alfabetul englez, 52 de caractere)
  - Cifre arabe (de la 0 la 9, 10 caractere)
  - caractere de spațiere (spatiu, tab, enter, etc)
  - caractere speciale ( . , ; = < > # \$ % + - \* / " ' ( ) etc)
  - alte caractere ( @ ` \$ etc.)
  - secvențe Escape / caractere speciale în C++:
    - \b Backspace, \t Tab orizontal, \v Tab vertical, \n Linie nouă, \f Pagina nouă – formfeed \r Început de rând, \" Ghilimele, \' Apostrof, \\ Backslash, \? Semnul întrebării, \a Alarmă
- Alte standarde de codificare: EBCDIC, ASCII, UTF-8, UTF-16.

# Unități lexicale

- Unități lexicale = atomii unui program C
  - **sintaxa** = reguli de îmbinare astfel încât să obținem programe corecte
  - unitățile lexicale sunt echivalente cu cuvintele unei limbi, iar *sintaxa* limbajului C cu *gramatica*
- Exemple de unități lexicale
  - comentarii
  - directive de preprocesare
  - valori constante
  - cuvinte cheie
  - identificatori
  - operatori
  - instrucțiuni

# Comentarii

- sunt secvențe din program ignorate de compilator.
  - Programatorul poate scrie aici observațiile sale privind părți din program.
- comentariile nu pot fi imbricate (i.e. continute unul in altul)
- comentariu pe o linie: începe cu `//` și se termină la sfârșitul liniei
- `// primul meu program C`
- `i++; // valoarea lui i crește cu 1`
  - comentariu bloc: încep cu `/*` și se termină la primul `*/`
    - este folosit pentru a nota o secvență mai mare/importantă din program
- `/*`
- `primul meu program C este`
- `programul Hello world`
- `*/`

# Exercițiu

- Găsiți comentariile corecte:

1. `// un exercitiu pentru voi //`
2. `/* un exercitiu pentru voi`
3. `// un exercitiu pentru voi */`
4. `// un exercitiu  
pentru voi`
5. `/* un exercitiu  
pentru voi  
*/`
6. `/**/`

# Cuvinte cheie

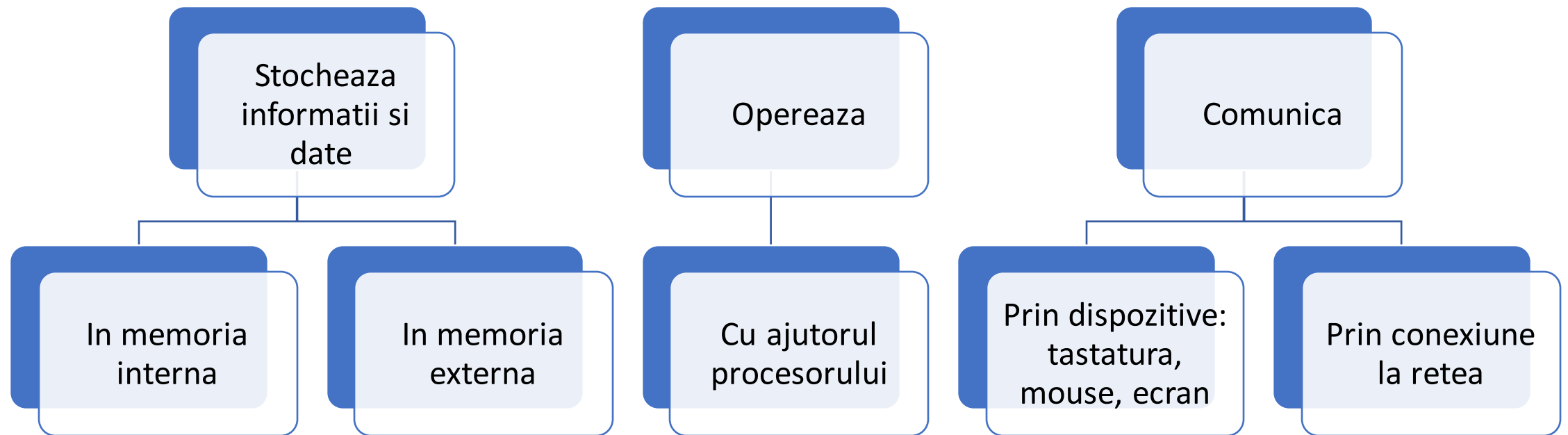
- secvență de caractere rezervate de limbaj
  - nu pot fi folosite ca identificatori

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

PROGRAM = DATE + OPERAȚII



# Ce fac computerele?



# Informații vs. date

## Informații – interpretarea unor date

- Numărul 123, Stringul „abc”

## Data – o colecție de simboluri stocate (într-o anumită reprezentare) într-un computer

- Informația: 12 – data(reprezentarea /codificarea) 1100
- Informația „abc” – data (reprezentarea/codificarea) 97 98 99 0

## Procesarea datelor și informațiilor

- Dispozitivele de intrare transformă informațiile în date
  - Datele sunt stocate în memorie
- Dispozitivele de ieșire produc informații din date

## Operații de bază ale procesoarelor

- În reprezentare binară
- Ex. AND, OR, NOT, XOR, etc.

# Tipuri de date

## Limbajul C este un limbaj tipizat

- fiecare informatie are asociat un tip de date, care determina:
  - Operatiile posibile
  - Semnificatia
  - Maniera de stocare

## Tipuri de date primare (de baza)

- caracter (char)
- număr întreg (int)
- număr real simplă precizie (float)
- număr real dublă precizie (double)

## Tipuri de date complexe (derivate)

- se obțin prin combinarea celorlalte tipuri.
- Ex. Agregate omogene (liste) de elemente de același tip
- Ex. Agregate neomogene de elemente de tipuri diferite care descriu un obiect din universul problemei de rezolvat

# Tipuri de date primare

Tip fundamentale	Mărime (octeți)	Domeniu de valori
char	1	Intre $-2^7$ (128) si $2^7 - 1$ (127) sau signed char: intre 0 ... 255
int	4 2	Intre $-2^{31}$ (-2.147.483.648) si $2^{32} - 1$ (2.147.483.647 ) Sau signed int: intre 0 si $2^{32} - 1$ -32768...32767
float	4	$-3,40 \times 10^38 \dots 3,4 \times 10^38$
double	8	$-1,79 \times 10^308 \dots 1,79 \times 10^308$

- Tipurile char și int admit subtipuri în funcție de care se modifică domeniul de valori
  - unsigned/signed **char**
  - short/long unsigned/signed **int**

# Reprezentarea in memorie - Example

```
char c = 'a'; // codul ASCII este  $97 = 2^6 + 2^5 + 2^0$ 
```

- In binar: 01100001 (primul bit = bitul de semn)

```
int x = 1;
```

- In binar, doar ultimul bit e 1: 00000000 00000000 00000000 00000001

```
float x = 0.5;
```

- $X = (-1)^{\text{semn}} \times \text{Mantissa} \times 2^{\text{Exponent}}$ 
  - 1 bit pt semn, exponentul este pe 8 biti, restul pana la 32 – mantissa

Detalii: <https://www.doc.ic.ac.uk/~eedwards/compsys/float/>

# Expresii

- În  $C$ , *orice* este o expresie.
- Reguli de bază:
  1. orice *constantă* este expresie;
  2. orice *variabilă* este expresie;
  3. dacă  $E$  este expresie, atunci și  $(E)$ ,  $-E$ ,  $+E$ ,  $f(E)$  sunt expresii, unde  $F$  este numele unei funcții aplicabile expresiei  $E$ ;
  4. dacă  $E_1$  și  $E_2$  sunt expresii, atunci și  $E_1 + E_2$ ,  $E_1 - E_2$ ,  $E_1 * E_2$ ,  $E_1 / E_2$  sunt expresii.

Vom vedea și alți operatori utilizați pentru formarea de expresii.

Constante

# Valori constante

- Constantă = informație care nu poate fi modificată de program
  - fiecare constantă are asociat un tip de date
- Tipuri de constante
  - caracter
  - șir de caractere
  - număr întreg
  - număr real



# Constante caracter

- un caracter ASCII scris între apostrofuri
  - 'C', '0', '!', '!'
  - au întotdeauna tipul char
- secvență escape = backslash (\) urmat de un caracter
  - cu semnificație specială
    - \' single quote – caracterul apostrof
    - \" double quote – caracterul ghilimele
    - \\ backslash – caracterul backslash
    - \n new line – trecerea la rândul următor
    - \t horizontal tab – tabulator orizontal (multiplu de 8)

# Constante șir de caractere

- o secvență de caractere scrise între ghilimele
  - "acesta este un sir"
  - "acest sir contine ghilimele astfel \" "
- se pot scrie pe mai multe randuri:
  - "acesta este un sir\  
pe mai multe randuri"
  - "acesta este un sir"  
"pe mai multe randuri"

# Constante număr întreg

- un număr întreg
  - scris în **baza 10** – exemplu: 12
    - folosind cifre 0...9
    - cifrele au ponderi egale cu puterea lui 10
  - scris în **baza 8** – exemplu: 014
    - folosind cifre 0...7
    - cifrele au ponderi egale cu puterea lui 8
  - scris în **baza 16** – exemplu: 0xC
    - folosind cifre 0...9 si litere A...F (sau a...f)
    - cifrele au ponderi egale cu puterea lui 16
- Operații cu întregi
  - + - \* / %
    - $a \% b$  este evaluata la restul împărțirii lui a la b
  - == != < <= > >=
  - ++ --

# Constante număr real

- un număr real scris în baza 10, cu sau fără zecimale, cu sau fără exponent
  - Exponentul = litera e urmată de un număr întreg n (puterea lui 10).
    - 10   10.0   10e0
    - 1.5   15e-1   0.15E1 =  $0.15 \times 10^1$
    - 0.3   .3   3E-1
  - separator pentru zecimale este punctul
- Operații cu numere reale
  - + - \* /
  - == != < <= > >=

# Constante logice

- In limbajul C, valorile logice de adevăr sunt codificate cu numere întregi, astfel: 0 – false, orice nr întreg nenul - true
- În C++ exista tipul bool care are domeniul de valori {false, true}, respectând convenția de mai sus

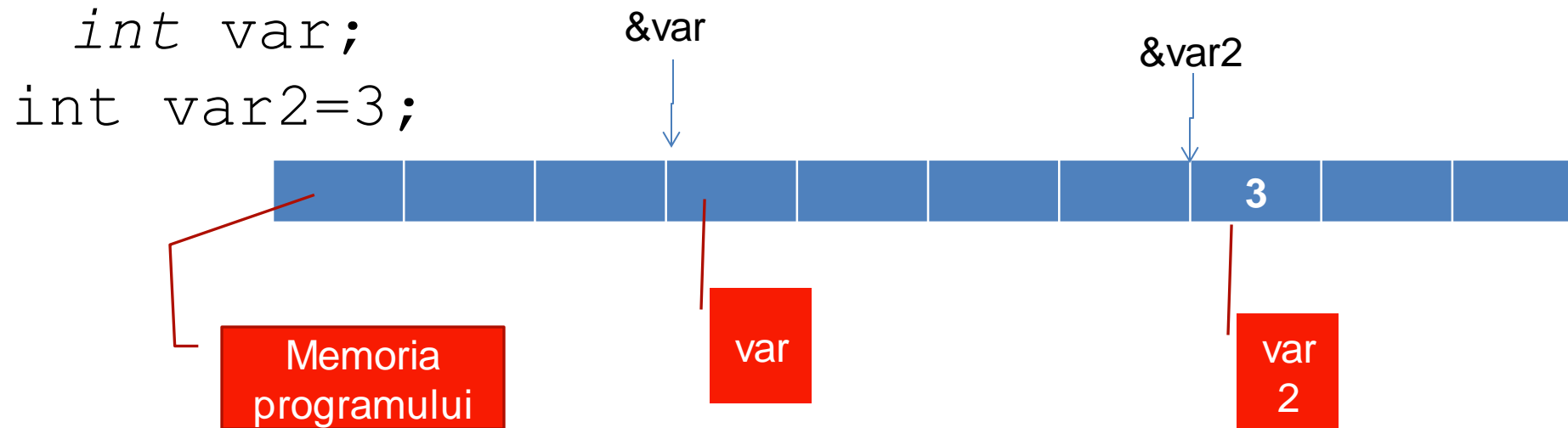
Operatii cu valori logice:

- And (&&),
- Or (||),
- Not (!)

Variabile

# Variabila

Variabila = o **zona de memorie** capabila sa retina o informatie de un anumit **tip**, accesibila programatorului printr-un **nume** si calculatorului, printr-o **adresa de memorie**



- un **nume** pe care programatorul îl dă unei entități (o variabilă, funcție, tip de date, etc) pentru a putea lucra mai ușor cu aceasta.
- identificator
  - secvență de caractere alfanumerice și underscore (`_`)
  - începe cu o literă sau underscore.
  - literele mari diferă de litere mici

# Declararea variabilei

- înainte de a fi folosită orice variabilă trebuie declarată
- `float PI = 3.1415926;`
- `int varsta = 26, greutate;`
- `greutate` este **declarata** (alocata fara valoare initiala)
- `PI` si `varsta` sunt **initializate** (alocate si cu valoare initiala)
- Tipul variabilei determină:
  - operațiile care se pot efectua asupra valorii variabilei
  - domeniul de valori a variabilei



# Atribuirea / asignarea

- Prin atribuire, o **variabila** primește valoarea unei **expresii**
- Sintaxa: **Variabila = expresie**
- Semantica: se evaluează *expresie*, se setează valoarea lui variabila la valoarea rezultată

Remarca.

În partea stângă a lui = se poate afla doar o variabilă.

În partea dreaptă a lui = se poate afla orice expresie.

Operatii de intrare/ieşire

# Afișarea datelor

- funcția **printf** afișează pe ecran un **mesaj** (șir de caractere)
  - mesajul poate să conțină coduri speciale pentru afișarea altor informații
    - %d, %i – informații cu tip întreg în baza 10
    - %o – întreg în baza 8, %x întreg în baza 16
    - %f – informații cu tip real (float sau double)
    - %c – date de tip caracter
    - %s – date de tip șir (vector) de caractere

```
int varsta = 20;  
float pi = 3.141528;  
printf("Varsta este %d", varsta);  
printf("Pi este aproximativ %f\n", pi);
```

# Citirea datelor

- funcția **scanf** citește de la tastatură un mesaj (șir de caractere)
    - mesajul poate să conțină coduri speciale pentru citirea unor informații în variabile
      - observați folosirea lui & înainte de numele variabilei
- ```
int varsta;  
printf("Ce varsta aveti?");  
scanf("%d", &varsta);
```
- scanf este alternativa pentru **cin>>variabila**.

# Directive de preprocesare

- **Preprocesare** = modificarea automată a programului sursă înainte de compilare
- Directivele de preprocesare încep cu #
  - #include, #define #ifdef etc.
  - */\* #include este înlocuită cu fișierul sursă indicat \*/*
  - #include <stdio.h>
  - */\* stdio.h contine functii pentru intrare/iesire \*/*

# Instructiunile *vida* si *compusa*

- instructiunea *vidă* nu execută nici o operație
  - se scrie doar ;
- Instructiunea *compusa* consta dintr-o succesiune de declaratii si instructiuni incluse intre acolade

```
{  
...//instructiuni simple  
}
```

Operatii

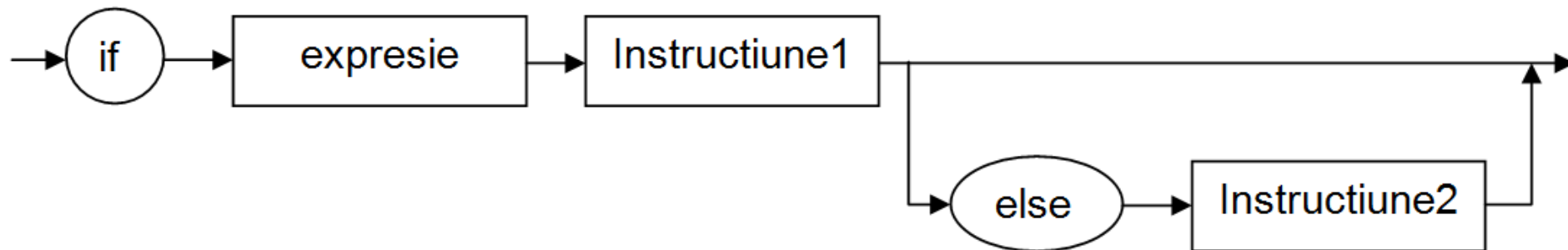
# Operatori aritmetici

- Operatori unari de păstrare/schimbare a semnului: + și –
- Operatori binari multiplicativi \*, / și %
- Operatori binari aditivi + și –
- Exemplu:
  - $\text{int } i = -2 + 3 * 4 - 5;$
  - Este diferit de  $-(2 + 3 * 4 - 5);$
  - Si este diferit de  $-2 + 3*(4 - 5);$



# Instrucțiunea decizionala **if**

- permite execuția unei instrucțiuni în funcție de valoarea unei expresii



- Ramura else și instrucțiune2 sunt opționale.

# Instrucțiunea decizionala **if** (cont)

- efect: se evaluează **expresie**;  
dacă rezultatul este nenul,  
atunci se execută **instrucțiune1**,  
altfel se execută **instrucțiune2**

- exemple:

```
if (delta < 0)
    printf("Delta este negativ\n");
else
    printf("Delta este 0 sau pozitiv\n");
```

```
if (bani > 100)
    bogat = 1; // oare?
```

# Totul depinde de “expresie”

- Operatori relationali (0=fals, nenul = adevarat)

< <= > >= == !=

- Atentie! `a==0` este diferit de `a=0`

- Operatori logici

! negatie logica

&& SI logic

|| SAU logic

# Negatia logica

- ! Este operator unar → are prioritatea cea mai ridicata

| X  | !X |
|----|----|
| ≠0 | 0  |
| 0  | 1  |

# Operatori logici

- && si || sunt operatori binari, au prioritatea mai mica decat operatorii relationali

| X  | Y  | X && Y |
|----|----|--------|
| 0  | 0  | 0      |
| 0  | ≠0 | 0      |
| ≠0 | 0  | 0      |
| ≠0 | ≠0 | 1      |

| X  | Y  | X    Y |
|----|----|--------|
| 0  | 0  | 0      |
| 0  | ≠0 | 1      |
| ≠0 | 0  | 1      |
| ≠0 | ≠0 | 1      |

# Legile lui De Morgan

- $\neg (A \ \&\& \ B)$  este echivalent cu  $\neg A \ || \ \neg B$
- $\neg (A \ || \ B)$  este echivalentă cu  $\neg A \ \&\& \ \neg B$

## Regula scurtcircuitului

- dacă primul operand al expresiei în care apare operatorul `&&` este 0, sigur rezultatul final este 0, indiferent de valoarea celui de-al doilea.
- dacă primul operand al expresiei în care apare operatorul `||` este `!=0`, sigur rezultatul final este 1, indiferent de valoarea celui de-al doilea
- Expresiile logice în C se calculează prin **scurtcircuitare**
  - dacă primul operand are valorile de mai sus, corespunzător operandului `&&` sau `||`, cel de-al doilea operand nu se mai evaluează.