Programarea calculatoarelor

#3

C++
Elemente introductive ale limbajului C++

Adrian Runceanu
www.runceanu.ro/adrian

2016

Curs 3
Elemente introductive ale limbajului C++

3. Elemente introductive ale limbajului C++

3.1. Programarea şi limbaje de programare

Prin **programare** se înţelege în mod generic *transpunerea unor operaţii repetitive, asupra unui set de date, într-un limbaj inteligibil de către un sistem de calcul care urmează ulterior să le execute*.

Acest lucru este realizat în două etape:

1. etapă în care este implicat omul şi anume cea de trecere de la problema reală la transpunerea într-un limbaj de programare.
2. o a doua etapă, automată, care transpune **codul sursă** (înşiruirea de instrucţiuni specifice limbajului respectiv) într-un **cod direct executabil** (inteligibil sistemului de calcul) lucru de care se ocupă programe specializate numite **compilatoare**.

3.1. Programarea şi limbaje de programare

Rolul programării este ca de fiecare dată când o anumită operaţiune sau o suită de operaţiuni repetitive care se aplică asupra unor seturi de date mereu diferite să fie scris un program care să:

1. ceară setul de **date de intrare** (cele care trebuie să fie prelucrate)
2. să execute asupra lor suita standard de operaţiuni
3. şi să livreze **datele de ieşire** (adică rezultatele)

3. Elemente introductive ale limbajului C++

3.2. Limbajul C

Limbajele de programare de nivel mediu au fost serios dezvoltate pe la mijlocul anilor '50.

La ora actuală se estimează că există peste 2000 de limbaje de programare, diferenţele între ele fiind legate în principal de stilul de programare.

Limbajul C, dezvoltat în 1972 de Dennis M. Ritchie* la Laboratoarele AT&T Bell, este primul limbaj pentru crearea de **sisteme de operare**.

**Dennis M. Ritchie a decedat pe 14.10.2011 (la varsta de 70 de ani)!*

3.2. Limbajul C

Numele limbajului provine din faptul că este rezultatul îmbunătăţirii limbajului B, folosit în scrierea sistemului de operare UNIX pentru DEC PDP7.

Prima documentaţie despre acest limbaj a fost "The C Programing Language", scrisă de Dennis Ritchie şi Brian Kernighan în 1977.

Înaintea ei exista doar "The C Reference Manual", scrisă de Dennis Ritchie.

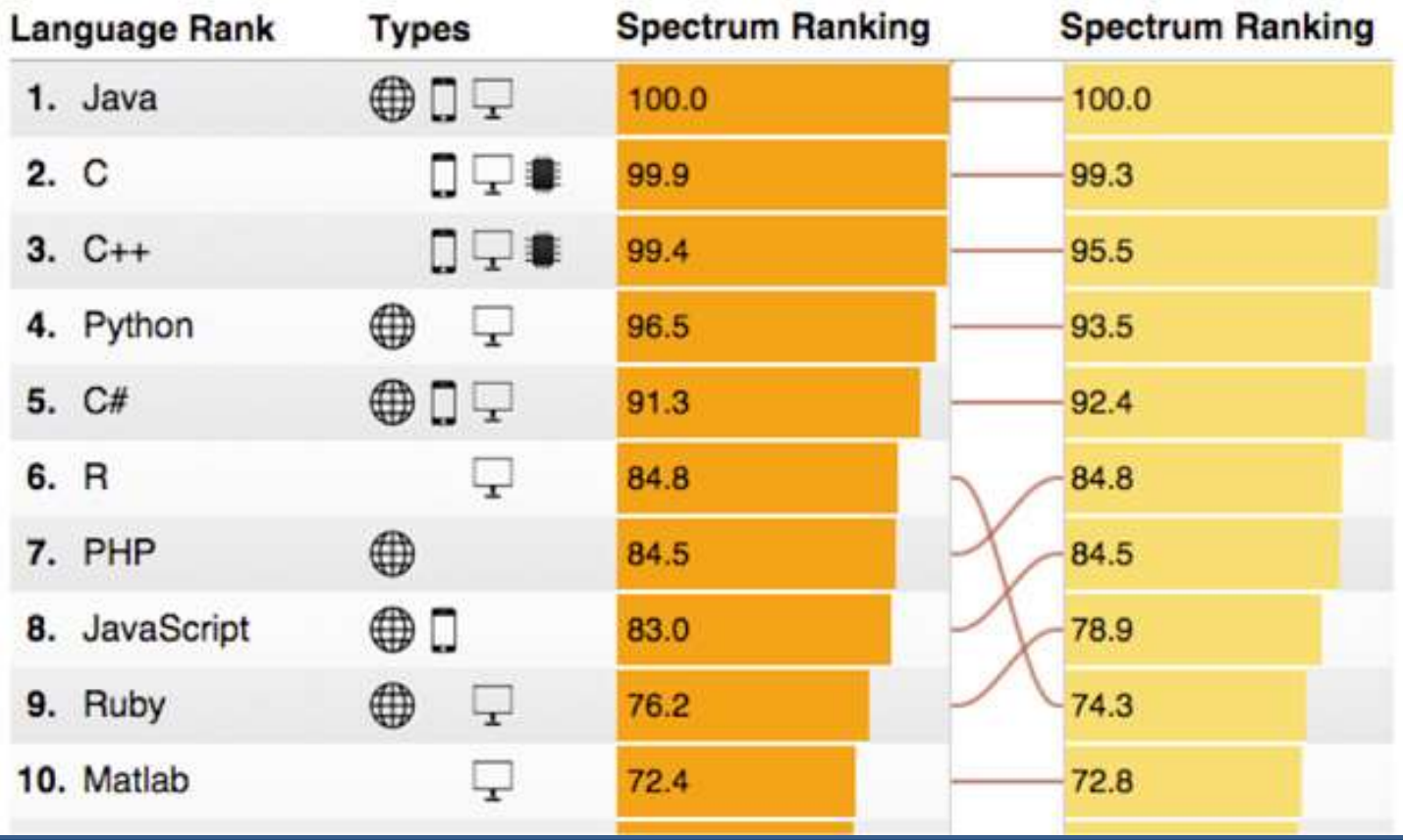
O caracteristică importantă a acestui limbaj este faptul că poate fi considerat simultan şi un limbaj de nivel mediu şi un limbaj de nivel scăzut.

3.2. Limbajul C

Limbajul C şi versiunile sale **OOP (Object Oriented Programming) C++, Visual C++** şi mai noul **C#** sunt printre cele mai folosite limbaje de programare la ora actuală.

3.2. Limbajul C

http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages



Language Rank	Types	Spectrum Ranking	Spectrum Ranking
1. Java		100.0	100.0
2. C		99.9	99.3
3. C++		99.4	95.5
4. Python		96.5	93.5
5. C#		91.3	92.4
6. R		84.8	84.8
7. PHP		84.5	84.5
8. JavaScript		83.0	78.9
9. Ruby		76.2	74.3
10. Matlab		72.4	72.8

3.2. Limbajul C

Limbajul C permite folosirea a două tehnici de programare:

1. programare structurată
2. acces direct la maşină

fapt care-l face să fie foarte flexibil.

Ultimul şi poate cel mai important motiv pentru învăţarea limbajului C este faptul că permite trecerea cu uşurinţă la varianta sa **C++,** la limbajul **Java** sau la limbajul **C#**.

3. Elemente introductive ale limbajului C++

3.3. Elemente de bază ale limbajului C++

Vom studia în cele ce urmează varianta orientată pe obiecte a limbajului standard C, şi anume **limbajul C++.**

3.3. Elemente de bază ale limbajului C++

1. Tipuri de date. Variabile. Constante

Cuvinte cheie

Limbajul C, ca orice limbaj de programare, este compus din câteva denumiri (identificatori) cu o semnificaţie bine stabilită, numite *cuvinte cheie.*

Observaţie: *Când alegeţi denumiri de variabile pentru programe să nu utilizaţi aceste denumiri.*

3.3. Elemente de bază ale limbajului C++

Cuvintele cheie ale limbajului C.

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*

Limbajul C++ adaugă noi cuvinte cheie la cele existente ale limbajului C.

asm	*bool*	*catch*	*class*	*delete*
friend	*inline*	*mutable*	*namespace*	*new*
operator	*private*	*public*	*protected*	*template*
this	*using*	*virtual*		

3. Elemente introductive ale limbajului C++

3.3.1. Tipuri de date

Un **tip de date** specifică (precizează):

- mulţimea de valori pe care variabila respectivă le poate lua
- cât şi setul de operaţii pe care programatorul le poate efectua cu acea variabilă

3.3.1. Tipuri de date

NUMELE TIPULUI	CARACTERISTICI
char	reţine un singur caracter Exemple: 'A', 'a', '%', etc.
int	reţine numere întregi cu semn Exemple: 23, -45, 0, etc.
float	reţine numere reale în format cu virgulă mobilă, în simplă precizie Exemple: 7.8965, -4.123, 7.0, etc.
double	reţine numere reale în format cu virgulă mobilă, în dublă precizie Exemple: 123456789.89654321, -123456789.1234567890, 123456789.0, etc. (se utilizează când se prelucrează numere foarte mari sau foarte mici)
void	tip de date special care nu specifică un anumit set de valori iniţial, dar care poate fi specificat ulterior declarării.

3.3.1. Tipuri de date

Reprezentarea caracterelor in memoria calculatorului

- ✓ Programatorii pot folosi in programe valori de orice tip (cifra, litera mica, litera mare, caractere speciale, alte caractere).
- ✓ In memoria calculatorului fiecare astfel de caracter se reprezinta printr-un cod numeric - ASCII (codul numeric al caracterului respectiv). Intervalul este intre 0 si 255.

Interval valori (selecție) Cod ASCII	Semnificație
[0, 32]	Caractere neprintabile(netipăribile)
[48, 57]	Cifrele de la 0 la 9
[65, 90]	Literele mari de la A la Z
[97, 122]	Literele mici de la a la z

3.3.1. Tipuri de date

1. Tipul de date char

char <definitie_de_data>;

Se reprezintă în memoria calculatorului folosind 8 biţi (un octet) şi poate păstra valori cuprinse între –128 şi 127.
Dacă se declară fără semn (adică se utilizează modificatorul *unsigned*), intervalul de valori se întinde de la 0 la 255.

Programatorii pot atribui valori de tip caracter unei astfel de variabile în două modalităţi distincte, dar care acţionează identic:

- ✓ reprezentarea din ASCII (codul numeric al caracterului respectiv)
- ✓ sau caracterul respectiv între două apostrofuri

Exemplu:

```
char litera_mica;
char litera_mica=97;
sau char  litera_mica='a';
```

3.3.1. Tipuri de date

2. Tipul de date int

```
int <definitie_de_data>;
```

Se reprezintă în memoria calculatorului folosind 16 biţi (2 octeţi) şi poate păstra valori cuprinse între –32768 şi 32767.

Dacă se declară fără semn (adică se utilizează modificatorul *unsigned*), intervalul de valori se întinde de la 0 la 65535.

Exemplu:

```
int a=9;
int b=6725;
int c=-31567;
```

3.3.1. Tipuri de date

3. Tipul de date float

```
float <definitie_de_data>;
```

Se reprezintă în memoria calculatorului folosind 32 biţi (4 octeţi) şi poate păstra valori cuprinse între 3.4E-38 şi 3.4E+38.

Exemplu:

```
float x=9.789;
float y=-6725.123;
float z=-3156723;
```

3.3.1. Tipuri de date

4. Tipul de date double

```
double <definitie_de_data>;
```

Se reprezintă în memoria calculatorului folosind 64 biţi (8 octeţi) şi poate păstra valori cuprinse între 1.7E-308 şi 1.7E+308.

Exemplu:

```
double numar_foarte_mare=12345678912345678.12345678912345678;
double numar_foarte_mic=-12345678912345678.12345678912345678;
double numar_mare=-123456789;
```

3.3.1. Tipuri de date

5. Tipul de date void

[void] <definitie_de_functie([void])

sau

void <definitie_de_pointer>;

- ✓ Este tipul de dată vidă (fără tip specificat), utilizat în general pentru mărirea clarităţii programelor.
- ✓ Tipul void permite explicitarea faptului că *o funcţie nu returnează nimic sau nu are nici un parametru*.

Exemplu:

```
void salut(void)
{
      cout<<"SALUTAM PROGRAMATORII IN LIMBAJUL C++ !!!\n";
}
```

3. Elemente introductive ale limbajului C++

3.3.2. Modificatorii de tip

Limbajul C++ oferă pe lîngă cele 5 tipuri de bază prezentate mai sus, un set de modificatori de tip:

1. ***unsigned*** (fără semn)
2. ***long*** (lung)
3. ***signed*** (cu semn)
4. ***register*** (registru)
5. ***short*** (scurt)

Exemplu: **unsigned int** numar;
register int i;
long int numar_foarte_mare;

- ✓ Un ***modificator de tip*** schimbă domeniul valorilor pe care o variabilă le poate păstra, sau modul în care compilatorul păstrează o variabilă.
- ✓ Pentru a se modifica un tip de data, *se va plasa modificatorul în fața tipului respectiv.*

3. Elemente introductive ale limbajului C++

3.3.3. Constante

Sunt date a căror valoare nu poate fi modificată în timpul execuţiei programului.

Ele reprezintă un tip şi o valoare şi astfel pot fi de mai multe tipuri:

1. **constantă întreagă**
2. **constantă flotantă**
3. **constantă caracter**
4. **constantă șir de caractere**

3.3.3. Constante

1. constantă întreagă = se reprezintă sub forma unei înşiruiri de cifre.

Se clasifică în:

- constante zecimale (se scriu în baza 10) Exemplu: 14, 568, 17342
- constante octale (se scriu în baza 8) Exemplu: **0**şir de cifre în baza 8
- constante hexazecimale (se scriu în baza 16) Exemplu: **0x** şir de cifre în baza 16

Constantele întregi se reprezintă pe 16 biţi sau pe 32 de biţi. Dacă la sfârşitul unei constante punem litera l sau L, atunci constanta respectivă va fi reprezentată pe 32 de biţi.

Exemplu: numărul 17 se reprezintă pe 16 biţi
numărul 17L se reprezintă pe 32 biţi

3.3.3. Constante

2. constantă flotantă este compusă din 2 părţi
– *partea fracţionară* (care poate fi vidă) şi
– *exponent* (care poate fi el vid)

O constantă reală este sub următoarea formă:

parte întreagă.parte fracţionară e ± exponent

Exemplu: 3.45e-17 ⇔ $3,45*10^{-17}$

Toate constantele flotante se reprezintă pe 16 biţi.

3.3.3. Constante

3. constantă caracter este de fapt un caracter între apostrofuri.

Se reprezintă pe 8 biţi, fiind chiar reprezentarea în codul ASCII a caracterului respectiv.

Exemplu:

'A' reprezentare internă: 65 (codul ASCII a caracterului 'A')

'a' reprezentare internă: 97 (codul ASCII a caracterului 'a')

În plus avem o notaţie specială '\' = backslash, care se poate folosi împreună cu câteva litere mici cu următoarele semnificaţii:

Caracter	semnificatie
\n	linie noua
\r	retur de car
\t	tabulator orizontal
\v	tabulator vertical
\\	backslash
\nnn	valoare ASCII in octal
\xnnn	valoare ASCII in hexazecimal

3.3.3. Constante

4. constantă şir sau şir de caractere

- ✓ Acest tip de constantă apare ca *o succesiune de caractere scrise între ghilimele.*
- ✓ Poate fi şi şirul vid. Reprezentarea internă este astfel încât fiecare caracter apare pe câte un singur octet, iar ca terminator de şir avem caracterul 0 (nul).
- ✓ Constantele şir pot fi scrise pe linii diferite, dar pe prima linie ultimul caracter este backslash, înainte de apăsarea tastei RETURN.

Exemplu: linia 1 : "conti\
linia 2 : nuare"

Exemplu: "AbbA" se reprezintă intern astfel:

659898650 A b b A

3. Elemente introductive ale limbajului C++

3.3.4. Variabile

- Pentru a putea utiliza informaţiile ce pot fi prelucrate prin intermediul programelor, trebuie să folosim denumiri (identificatori), care să fie compuşi din caractere – litere, cifre şi liniuţa de subliniere - **underscore('_')** din maximum 31 caractere.

- Numim **variabilă** *o denumire (identificator) pe care compilatorul o asociază cu o anumită zonă de memorie*.

3.3.4. Variabile

Când se declară o variabilă, trebuie specificat atât numele ei cât şi tipul de date asociat.

Exemple:

```
int variabila_de_tip_intreg;
float variabila_de_tip_real;
char variabila_de_tip_caracter;
void variabila_fara_tip;
```

Restricţie: *Numele variabilelor nu poate să înceapă cu o cifră.*

Exemplu: variabila1 - este corect

1variabila - nu este corect

3.3.4. Variabile

Observaţie:

- Limbajul C este *case sensitive*, adică *face diferenţa dintre literele mici şi mari*, astfel încât, două denumiri de variabile sau de funcţii, care sunt identice dar sunt scrise o dată cu litere mici iar apoi cu litere mari, se consideră ca fiind două denumiri de variabile sau de funcţii diferite.

Exemplu: int var_intreaga;
int VAR_INTREAGA;
semnifică două denumiri total diferite.

3.3.4. Variabile

Variabilele pot fi:

1. simple

2. compuse:
    a) tablou
    b) structură/uniune/enumerare

3.3.4. Variabile

1. Variabilele simple

Declaraţia de variabilă simplă are forma:

tip nume_variabila;

Exemplu:

```
int i;
int j, k, l;
double a, b;
float x, y;
char m, n, t;
```

3.3.4. Variabile

2. Variabilele tablou

Prin tablou înţelegem o *mulţime ordonată de acelaşi tip*; accesul la elementele tabloului făcându-se cu ajutorul indicilor.

Declaraţia este:

```
tip nume_tablou[dimensiune];
```

Exemplu:

```
int v[5];
float x[15];
double a[3];
```

3.3.4. Variabile

Observaţie:

Numerotarea elementelor unui tablou în limbajul C++ începe cu indicele 0.

Elementele lui int v[5] vor fi:

v[0],v[1],v[2],v[3],v[4];

Indice poate să fie orice expresie întreagă.

Putem avea chiar şi tablouri de şiruri de caractere: char t[20];

Numele tabloului este de fapt adresa primului său element.

3.3.4. Variabile

Iniţializarea variabilelor

Poate fi făcută chiar pe linia de declarare a variabilelor:

Exemplu:

```
int i=5;
float x=7.8;
int v[5]={1,2,7,10,-5};
float y[3]={-9.034,89,2};
char c='B';
```

Pentru iniţializarea variabilelor de tip şir de caractere avem următoarele posibilităţi:

```
char t[15] = { 's', 'i', 'r', ' ', 'c', 'o', 'r', 'e', 'c', 't', '\0' };
```

sau

```
char t[15] = "sir corect";
```

3.3.4. Variabile

Comentarii în programe

- ✓ Numim **comentarii,** *acele texte care nu sunt luate în considerare de compilator şi care apar între simbolurile*

/ *comentariu* **/*

sau

când este vorba despre o singură linie

// comentariu

- ✓ Se mai pot pune comentarii pentru ca să se elimine una sau mai multe instrucţiuni din programul C++.

3. Elemente introductive ale limbajului C++

3.4. OPERATORII LIMBAJULUI C++

Expresii

- ✓ O expresie poate să fie un operand sau mai mulţi operanzi legaţi prin operatori
- ✓ *Orice expresie are tip şi valoare care se obtin după evaluarea expresiei*

3.4. OPERATORII LIMBAJULUI C++

Operatori

- ✓ Operatorii folosiţi în limbajul C++ au o asociere de la stânga la dreapta – în general – cu excepţia
  - *operatorilor unari (se aplică la un singur operand),*
  - *relaţionali*
  - *şi de atribuire,*
- ✓ la care *asocierea se face de la dreapta la stânga*.

3.4. OPERATORII LIMBAJULUI C++

Operatorii sunt împărţiţi în 11 categorii:

	Operatori
1	aritmetici
2	relaţionali
3	de egalitate
4	logici
5	logici pe biţi
6	de atribuire
7	de incrementare şi decrementare
8	de conversie explicită (cast)
9	de lungime (sizeof)
10	condiţional
11	virgulă

3. Elemente introductive ale limbajului C++

3.4.1. Operatori aritmetici

OPERATOR	FUNCŢIE
+	Adunare
-	Scădere
*	Înmulţire
/	Împărţire
%	Restul împărţirii
+	adunare unară
-	scădere unară

În cele mai simple programe se pot utiliza operaţii matematice cum ar fi adunarea, scăderea, înmulţirea şi împărţirea.

Exemplu:

```
int i=9, j=2;
```

atunci **i/j** are ca rezultat 4

i%j are ca rezultat 1

3.4.1. Operatori aritmetici

Prezentăm în următorul program scris în C++, principalii operatori matematici:

```
#include <iostream.h>
int main(void)
{
 int secunde_pe_ora;
 float media;
 secunde_pe_ora = 60 * 60;
 media = (5 + 10 + 15 + 20) / 4;
 cout<<“Numarul de secunde intr-o ora este ”<< secunde_pe_ora
   <<endl;
 cout<<“Media numerelor 5, 10, 15 si 20 este ”<<media<<endl;
 cout<<“Numarul de secunde in 48 de minute este
   ”<<secunde_pe_ora – 12 * 60<<endl;
}
```

3.4.1. Operatori aritmetici

După execuţia programului se vor afişa pe ecran următoarele rezultate:

Numarul de secunde intr-o ora este 3600
Media numerelor 5, 10, 15 si 20 este 12.000000
Numarul de secunde in 48 de minute este 2880

3. Elemente introductive ale limbajului C++

3.4.2. Operatori relaţionali

În programe, prin aplicarea acestor operatori relaţionali se pot obţine două valori posibile, la evaluarea expresiilor care îi conţin:

0 – ceea ce înseamnă că expresia este falsă

1 – ceea ce înseamnă că expresia este adevărată

OPERATOR	FUNCŢIE
<	mai mic
<=	mai mic sau egal
>	mai mare
>=	mai mare sau egal

Exemplu:

```
int i=3, j=8;
```

Atunci pentru expresia **i < j** avem valoarea 1

Iar pentru expresia **i >= j** avem valoarea 0

3.4.2. Operatori relaţionali

Se citesc doua numere întregi a si b. Să se realizeze un algoritm care să verifice care numar este mai mare, afişandu-se un mesaj corespunzator:

```
#include <iostream.h>
int main(void)
{
    int a,b;
    cin>>a; cin>>b;
    if(a > b)
        cout<<“Numarul a este mai mare decat numarul b\n”;
    else
        cout<<“Numarul b este mai mare decat numarul a\n”;
}
```

3. Elemente introductive ale limbajului C++

3.4.3. Operatori de egalitate

În programe, prin aplicarea acestor operatori de egalitate se pot obţine două valori posibile, la evaluarea expresiilor care îi conţin:

0 – ceea ce înseamnă că expresia este falsă

1 – ceea ce înseamnă că expresia este adevărată

OPERATOR	FUNCŢIE
==	egal
!=	diferit

Exemplu:

int i=2, j=5, k=2;

Atunci pentru expresia **i!=j** avem valoarea 1

Pentru expresia **i==j** avem valoarea 0

Iar pentru expresia **i==k** avem valoarea 1

3.4.3. Operatori de egalitate

Se citeşte un număr întreg a. Să se realizeze un algoritm care să verifice dacă numărul a este par, afişandu-se un mesaj corespunzator:

```
#include <iostream.h>
int main(void)
{
    int a;
    cin>>a;
    if( a % 2 == 0 )
        cout<<“Numarul este par\n”;
    else
        cout<<“Numarul este impar\n”;
}
```

Referinte bibliografice

Bibliografia necesară cursului:

1. **Adrian Runceanu**, Mihaela Runceanu, **Noțiuni de programare în limbajul C++**, Academica Brâncuşi, Târgu-Jiu, 2012, ISBN 978-973-144-550-2, 483 pagini
2. **Adrian Runceanu**, **Programarea şi utilizarea calculatoarelor**, Editura Academică Brâncuși Targu-Jiu, 2003
3. Octavian Dogaru, **C++ - Teorie şi practică**, volumul I, Editura Mirton, Timişoara, 2004
4. O.Catrina, I.Cojocaru, *Turbo* C+, Editura Teora, Bucureşti, 1993
5. D.Costea, *Iniţiere în limbajul* C, Editura Teora, Bucureşti, 1996
6. K.Jamsa, *C++,* Editura Teora,1999
7. K.Jamsa & L.Klander, *Totul despre C si C++,* Teora, 2004

Întrebări?