Universitatea Politehnica Bucuresti

Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei

Portabilitatea limbajului de programare C/C++

Florescu Iulian - Cosmin

432A

Cuprins

1. Introducere

Ce este limbajul de programare C ? Limbajul "C" este un limbaj de programare care are o destinatie universala. El este utilizat in rezolvarea problemelor stiintifice si tehnico-ingineresti, in prelucrari de date, precum si in scrierea programelor de sistem, este un limbaj de programare standardizat, compilat, de nivel mediu. Este implementat pe majoritatea platformelor de calcul existente azi, si este cel mai popular limbaj de programare pentru scrierea de software de sistem. Este apreciat pentru eficienta codului obiect pe care il poate genera, si pentru portabilitatea sa.

A fost dezvoltat la inceputul anilor 1970 de Ken Thompson si Dennis Ritchie, care aveau nevoie de un limbaj simplu si portabil pentru scrierea nucleului sistemului de operare UNIX pe minicalculatoarele firmei DEC, seria PDP-11.

C este un limbaj de programare relativ minimalist ce opereaza in stransa legatura cu hardware-ul, fiind cel mai apropiat de limbajul de asamblare fata de majoritatea celorlalte limbaje de programare.

Scurta introducere in C

C a fost creat avand drept scop important de a face ca programele mari sa poata fi scrise mai usor si cu mai putine erori in paradigma programarii procedurale, dar fara a pune un obstacole in scrierea compilatorului de C, care este incarcat de caracteristicile complexe ale limbajului. C are urmatoarele caracteristici importante:

- Este un limbaj de baza simplu, cu importante functionalitati cum ar fi functiile matematice sau cele de manipulare ale fisierelor
- Este focalizat pe paradigma programarii procedurale, care faciliteaza programarea intr-un mod structurat
- Utilizeaza un set simplu de tipuri de date ce impiedica multe operatii neintentionate
- Foloseste un limbaj preprocesor, preprocesorul C, pentru sarcini cum ar fi definirea de macrouri si includerea mai multor fisiere sursa
- Permite accesarea la nivel scazut a memoriei calculatorului prin utilizarea pointerilor
- Permite folosirea parametrilor, care sunt comunicati functiilor prin valoare si nu prin referinta
- Pointeri la functii, ce permit forme rudimentare de inchidere (engleza *closure*) si polimorfism
- Declararea variabilelor
- Structuri de date sau tipuri de date agregate, definite de utilizator prin (struct), ce permit ca date inrudite sa fie combinate si manipulate ca un intreg.

Ce este limbajul de programare C++ ? **C++** este un limbaj de programare general, compilat. Este un limbaj multi-paradigma, cu verificarea statica a tipului variabilelor ce suporta programare procedurala, abstractizare a datelor, programare orientata pe obiecte. In anii 1990, C++ a devenit unul din cele mai populare limbaje de programare comerciale, ramanand astfel pana azi.

Bjarne Stroustrup de la Bell Labs a dezvoltat C++ (initial denumit *C cu clase*) in anii 1980, ca o serie de imbunatatiri ale limbajului C. Acestea au inceput cu adaugarea notiunii de clase, apoi de functii virtuale, suprascrierea operatorilor, mostenire multipla, sabloane (engleza *template*) si exceptii. Limbajul de programare C++ a fost standardizat in 1998 ca si ISO 14882:1998, versiunea curenta fiind din 2003, ISO 14882:2003. Urmatoarea versiune standard, cunoscuta informal ca C++0x, este in lucru.

Istoricul C++

Bjarne Stroustrup a inceput sa lucreze la C cu clase in 1979. Ideea crearii unui nou limbaj a venit din experienta de programare pentru pregatirea tezei sale de doctorat. Stroustrup a descoperit ca Simula avea facilitati foarte utile pentru proiecte mari, insa era prea lent, in timp ce BCPL era rapid, insa nu era de nivel inalt si era nepotrivit pentru proiecte mari. Cand a inceput sa lucreze pentru Bell Labs, avea sarcina de a analiza nucleul UNIX referitor la calcul distribuit. Amintindu-si de experienta sa din perioada lucrarii de doctorat, Stroustrup a inceput sa imbunatateasca C cu facilitati asemanatoare Simula. C a fost ales deoarece era rapid si portabil. La inceput facilitatile adaugate C-ului au fost clase, clase derivate, verificare a tipului, inline si argumente cu valori implicite.

In 1982, numele limbajului a fost schimbat de la C cu clase la C++. Au fost adaugate noi facilitati, inclusiv functii virtuale, supraincarcarea operatorilor si a functiilor, referinte, constante, alocare dinamica, un control al tipului mai puternic si noua varianta de comentariu pe un singur rand (liniile care incep cu caracterele '//').

In 1985 a fost lansata prima editie a cartii "The C++ Programming Language" (Limbajul de programare C++), oferind informatii importante despre limbaj, care inca nu era un standard oficial. In 1989 a fost lansata versiunea 2.0 a C++. Au aparut acum mostenirea multipla, clase abstracte, functii statice, functii constante si membri protected. In 1990 o alta carte a fost lansata, oferind suport pentru standarde viitoare. Ultimele adaugari includeau template-uri, exceptii, spatii de nume (namespace-uri) si tipul boolean.

O data cu evolutia limbajului C++, a evoluat si o biblioteca standard. Prima adaugire a fost biblioteca de intrari/iesiri (I/O stream), care oferea facilitati pentru a inlocui functiile traditionale C cum ar fi printf si scanf. Mai tarziu, printre cele mai semnificative adaugari la biblioteca standard a fost STL (Standard Template Library) (Biblioteca de formate standard).

Dupa ani de lucru, un comitet ANSI-ISO a standardizat C++ in 1998 (ISO/IEC 14882:1998).[1]

2. Etapele rezolvarii unei problem in limbajul C/C++

Ciclul de dezvoltare al unui program contine urmatoarele etape:

1. **Definirea problemei de rezolvat.**

Analiza problei cuprinde in afara formularii problemei in limbaj de natural, o precizare riguroasa a intrarilor (datelor problemei) si a iesirilor (rezultatelor).

De exemplu ne propunem sa rezolvam ecuatia de gradul 2: $ax^2+bx+c=0$

Datele de intrare sunt cei 3 coeficienti a, b, c ai ecuatiei, care precizeaza o anumita ecuatie de grad 2.

Rezultatele sunt cele doua radacini (reale sau complexe) sau celelalte situatii particulare care pot apare.

2. **Identificarea pasilor necesari pentru rezolvarea problemei** incepe cu formularea ***modelului matematic*.**

Ca model matematic vom folosi formula:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Formula nu este intotdeauna aplicabila. Vom distinge urmatoarele situatii:

1. a=0, caz in care nu avem de a face cu o ecuatie de gradul 2, ci
   1.1. este posibil sa avem ecuatia de gradul 1: bx+c=0, cu solutia x= - c/b , daca b ≠ 0.
   1.2. daca si b=0, atunci
      1.2.1. daca c≠0, atunci nu avem nici o solutie, in timp ce
      1.2.2. daca si c=0, atunci avem o infinitate de solutii.
2. a≠0 corespunde ecuatiei de gradul 2. In acest caz avem alte doua situatii:
   2.1. Formula este aplicabila pentru radacini reale (discriminant pozitiv)
   2.2. Pentru discriminant negativ, intrucat nu dispunem de aritmetica complexa, va trebui sa efectuam separat calculele pentru partea reala si cea imaginara.

3. Proiectarea algoritmului folosind ca instrument **pseudocodul.**
Pseudocodul folosit cuprinde:

- operatii de intrare / iesire:

citeste var_1, var_2,...
scrie $expresie_1$, $expresie_2$,...

- structuri de control:
  - decizia:

daca expresie atunci
  $instructiune_1$;
altfel
  $instructiune_2$;

  - ciclul:

cat timp expresie repeta
  instructiune;

  - secventa:

{ $instructiune_1$;
  ...
  $instructiune_n$;
}

Algoritmul dezvoltat pe baza acestui pseudocod este:

```
reali a,b,c,delta,x1,x2,xr,xi;
citeste a,b,c;
daca a=0 atunci
  daca b≠0 atunci
    scrie –c/b;
  altfel
    daca c≠0 atunci
          scrie “nu avem nici o solutie”;
    altfel
          scrie “o infinitate de solutii”;
  altfel
    { delta=b*b-4*a*c;
```

```
    daca delta >= 0 atunci {
          x1=(-b-sqrt(delta))/(2*a);
          x2=(-b+sqrt(delta))/(2*a);
          scrie x1,x2;
         }
         altfel {
          xr=-b/(2*a);
          xi=sqrt(-delta)/(2*a);
          scrie xr,xi;
  }
}
```

4. Scrierea programului folosind un limbaj de programare.

Vom codifica algoritmul descris mai sus folosind limbajul C:

```
#include <stdio.h>
#include <math.h>
void main(void)
{ double a,b,c,delta,x1,x2,xr,xi;
  scanf("%f %f %f",&a,&b,&c);
  if (a==0)
    if (b!=0)
      printf("o singura radacina x=%6.2f\n",-c/b);
    else
      if (c!=0)
        printf("nici o solutie\n");
      else
        printf("o infinitate de solutii\n");
  else
    { delta=b*b-4*a*c;
      if(delta >= 0)
        { x1=(-b-sqrt(delta))/2/a;
          x2=(-b+sqrt(delta))/2/a;
          printf("x1=%5.2f\tx2=%5.2f\n",x1,x2);
        }
      else
        { xr=-b/2/a;
          xi=sqrt(-delta)/2/a;
          printf("x1=%5.2f+i*%5.2f\nx2=%5.2f-i*%5.2f\n",
                xr,xi,xr,xi);
        }
    }
}
```

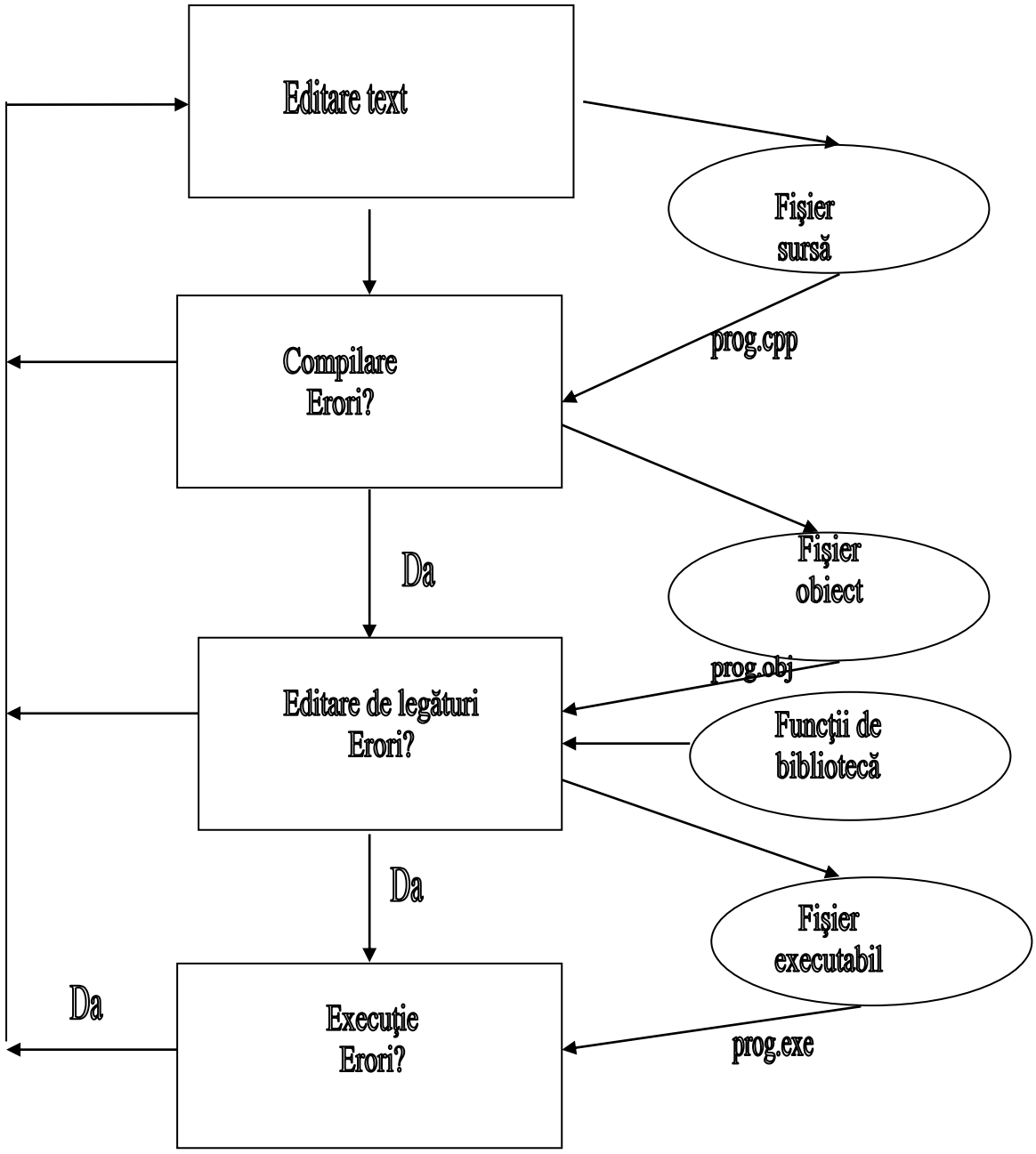


Fig.1. Etapele rezolvarii unei probleme folosind calculatorul

5. Implementarea programului: editare, compilare, editare de legaturi, executie.

Sa detaliem in continuare ***etapa de implementare***. Dupa analiza problemei si stabilirea algoritmului, acesta trebuie tradus (**implementat**) intr-un limbaj de programare.

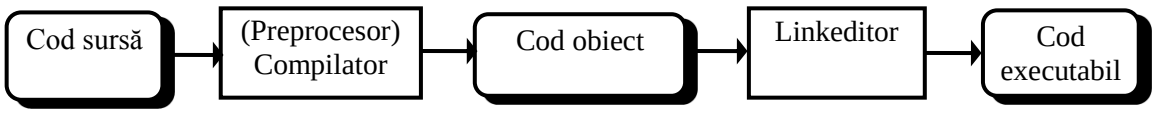
- Scrierea (editarea) programului sursa cu ajutorul unui editor de text
  Programele sursa sunt fisiere text care contin instructiuni (cu sintactica si semantica proprii limbajului utilizat). Programul (fisierul) sursa este creat cu ajutorul unui *editor de texte* si va fi salvat pe disc (programele sursa C primesc, de obicei, extensia *.c*, iar cele C++, extensia ***.cpp*)**.
  Pentru a putea fi executat, programul sursa trebuie *compilat* si *linkeditat*.

- Compilarea
  Procesul de compilare este realizat cu ajutorul compilatorului, care translateaza codul sursa in cod obiect (cod masina), pentru ca programul sa poata fi inteles de calculator. In cazul limbajului C, in prima faza a compilarii este invocat *preprocesorul*. Acesta recunoaste si analizeaza mai intai o serie de instructiuni speciale, numite *directive procesor*. Verifica apoi codul sursa pentru a constata daca acesta respecta

sintaxa si semantica limbajului. Daca exista erori, acestea sunt semnalate utilizatorului. Utilizatorul trebuie sa corecteze erorile (modificand programul sursa). Abia apoi codul sursa este translatat in cod de asamblare, iar in final, in cod masina, binar, propriu calculatorului. Acest cod binar este numit cod obiect si de obicei este memorat intr-un alt fisier, numit *fisier obiect*. Fisierul obiect va avea, de obicei, acelasi nume cu fisierul sursa si extensia *.obj*.

- Linkeditarea
  Dupa ce programul sursa a fost translatat in program obiect, el este va fi supus operatiei de linkeditare. Scopul fazei de linkeditare este acela de a obtine o forma finala a programului, in vederea executiei acestuia. Linkeditorul "leaga" modulele obiect, rezolva referintele catre functiile externe si rutinele din biblioteci si produce cod executabil, memorat intr-un alt fisier, numit *fisier executabil* (acelasi nume, extensia *.exe*)

- Executia
  Lansarea in executie consta in incarcarea programului executabil in memorie si startarea executiei sale.



6. Testarea si depanarea programului.

Daca s-a depasit faza erorilor la executie, vom testa programul, furnizandu-i date de intrare pentru care cunoastem iesirile. Aparitia unor neconcordante indica prezenta unor *erori logice*. Daca insa, rezultatele sunt corecte, nu avem certitudinea ca programul functioneaza corect in toate situatiile. Prin testare putem constata prezenta erorilor, nu insa si absenta lor.

Desfasurarea calculelor poate fi controlata prin executie pas cu pas,sau prin asigurarea unor puncte de intrerupere in care sa inspectam starea programului, folosind in acest scop un *depanator* (debugger).[2]

3. Ce inseamna portabilitatea C/C++ ?

Portabilitatea este caracteristica unei aplicatii de a putea fi folosita intr-un alt mediu in afara aceluia pentru care a fost initial proiectata. Acest lucru poate insemna un alt sistem de operare, o alta arhitectura hardware, o alta biblioteca. Se spune ca o aplicatie este portabila daca are nevoie de modificari minime pentru a putea rula pe un alt mediu. Actiunea de portare este actiunea de modificare a unei aplicatii pentru a putea fi folosita pe un alt sistem de operare sau alt sistem fizic. Un mediu pe care se realizeaza portarea se mai numeste platforma.

Portabilitatea unui limbaj de progamare

Programarea in limbaj de asamblare nu asigura, prin definitie, portabilitatea codului, deoarece acesta este intrinsec legat de arhitectura sistemului de calcul. Unul dintre motivele respandirii Unix in anii '70 a fost scrierea acestuia in C, un limbaj de nivel inalt portabil. Folosirea C inseamna eliberarea de arhitectura hardware pe care va rula aplicatia. Portarea codului era facuta prin intermediul compilatorului. In ziua de azi, aplicatiile C sunt cele mai usor portabilie datorita existentei unui compilator C pe orice platforma.

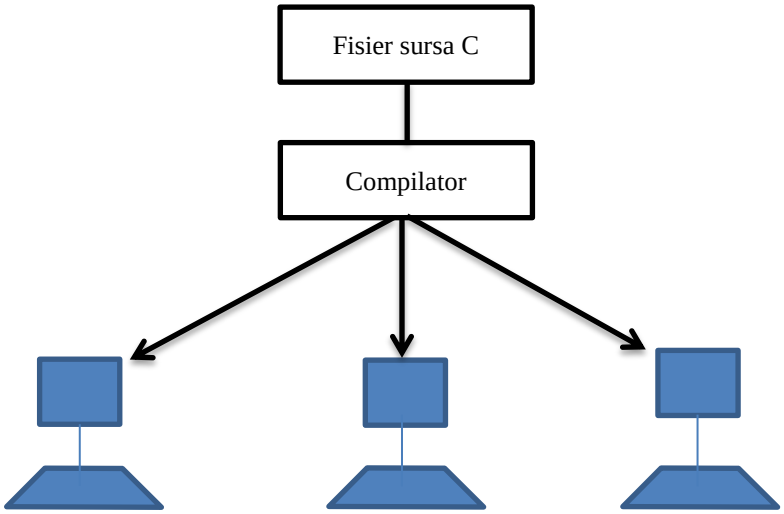


Fig 3. Portabilitatea asigurata de compilator

Portabilitatea C la nivelul sistemului de operare

Portabilitatea unei aplicatii se poate referi la posibilitatea rularii acesteia pe un alt sistem de operare. Acest lucru este corelat, de obicei, cu interfetele de programare (API – Application Programming Interface) pe care sistemul de operare le pune la dispozitie (system API). Astfel, Windows pune la dispozitie programatorului C interfata Win32API. Unix, pe de alta parte pune la dispozitia programatorului interfata POSIX (Portable Operatin System Interface). Pentru deschiderea unui fisier un programator Windows va folosi CreateFile, iar un programator Unix open.

Pentru asigurarea portabilitatii insa, multe biblioteci ofera programatorului o interfata protabila peste sistemul de operare. Astfel, functiile ANSI din biblioteca standard C sunt portabile peste diverse sisteme de operare. Astfel, pentru ca deschiderea unui fisier, programatorul va folosi *fopen* indiferent de sistemul de operare pe care va rula aplicatia.[3]

C este prezentat uneori ca "asamblor portabil", facandu-se astfel diferentele principale fata de limbajele de asamblare: codul unui program C poate fi compilat si rulat pe aproape orice tip de masina (calculator), asemanator altor limbaje de programare, in timp ce limbajele de asamblare sunt specifice unui anumit model de masina. Limbajul C apartine clasei limbajelor de *nivel scazut* sau de *nivel mediu*, aceasta indicand stransa legatura intre interoperabilitate si echipamentul hardware.

Popularitatea limbajului a crescut rapid datorita elegantei si a multiplelor posibilitati oferite programatorului (puterea si flexibilitatea unui limbaj de asamblare); ca urmare, au aparut numeroase alte implementari. De aceea, in anii ’80 se impune necesitatea standardizarii acestui limbaj. In perioada 1983-1990, un comitet desemnat de ANSI (American National Standards Institute) a elaborat un compilator ANSI C, care permite scrierea unor programe care pot fi portate fara modificari, pe orice sistem.

Sistemul de operare UNIX, compilatorul C si in esenta toate aplicatiile sub sistemul UNIX sunt scrise in C intr-o proportie mare. Astfel, din cele 13000 linii sursa ale sistemului de operare UNIX, numai 800 linii sunt scrise in limbaj de asamblare, restul fiind scrise in C. De asemenea, insasi compilatorul C este scris in C in proportie de 80%. In felul acesta limbajul C asigura o portabilitate buna pentru programele scrise in el.

In prezent limbajul C este implementat si sub alte sisteme de operare. Practic el este disponibil pe toate calculatoarele, incepand cu microcalculatoarele personale si terminand cu

supercalculatoarele. Pe calculatoarele de tip IBM PC este implementata o varianta a limbajului C numita TURBO C. Aceasta varianta dispune de un mediu de programare menit sa ajute utilizatorul in scrierea si punerea la punct a programelor. De asemenea, pe acelasi tip de calculatoare este implementata varianta quickC, care dispune si ea de un mediu de programare dezvoltat.

Limbajul C contine structurile proprii programarii structurate. De asemenea, limbajul C dispune si de facilitati oferite de limbajele de asamblare, cum sunt lucrul pe biti si utilizarea adreselor.

El este considerat ca fiind un intermediar intre limbajele de nivel inalt si cele de asamblare. Compilatorul C ofera programatorului o flexibilitate mai mare in scrierea programelor decat alte limbaje de programare. El realizeaza un numar mai redus de controale la compilarea textului sursa. Din aceasta cauza programarea in limbajul C este mai expusa la erori decat programarea in alte limbaje, cum ar fi de exemplu, limbajul Pascal.

In limbajul C putem mentiona ca o cauza a reducerii portabilitatii ultizarea campurilor de biti, care impune unele restrictii:

- Tipul membrilor poate fi int sau unsigened int.
- Lungime este o constanta intreaga din intervalul [0, 31];
- Un camp de biti nu poate fi operandul unui operator de referentiere.
- Nu se pot organiza tablouri de campuri de biti.

Datorita restrictiilor pe care le impune folosirea campurilor de biti, cat si datorita faptului ca aplicatiile care folosesc astfel de structuri de date au o portabilitate extrem de redusa (organizarea memoriei depinzand de sistemul de calcul), se recomanda folosirea campurilor de biti cu precautie, doar in situatiile in care se face o economie substantiala de memorie. [2]

4. Bibliografie

1. http://en.wikiversity.org/wiki/Topic:C
   http://ro.wikipedia.org/wiki/C_%28limbaj_de_programare%29
   Doina Logofătu: *Bazele programarii in C. Aplicații*, Ed. 1, Editura Polirom, Iași, 2006, [1]

2. a) Brookshear, J.G., *Introducere in informatica*, Editura Teora, Bucuresti, 1998, b) Somnea, D., Turturea, D., *Initiere in C++ - Programarea orientate pe obiecte*, Editura Tehnica, Bucuresti 1993 c) Stroustrup, B., A Beginners C++, http://www.uow.edu.au/~nabg/ABC/C9.html [2]

3. Rughis R., Deaconescu R., Milescu G., Bardac M., *Utilizarea Sistemelor de Operare* http://books.google.ro [3]