Programarea calculatoarelor

#6

C++
Instrucţiunile limbajului C++ (continuare)

Adrian Runceanu
www.runceanu.ro/adrian

2016

Curs 6
Instrucţiunile limbajului C++ (continuare)

5. Instrucţiunile limbajului C++

5.1. Instrucţiunea **vidă**
5.2. Instrucţiunea **compusă**
5.3. Instrucţiunea **expresie**
5.4. Instrucţiunea **if**
5.5. Instrucţiunea **while**
5.6. Instrucţiunea **do while**
5.7. Instrucţiunea **for**
5.8. Instrucţiunea **switch**
5.9. Instrucţiunea **break**
5.10. Instrucţiunea **continue**
5.11. Instrucţiunea **goto**
5.12. Instrucţiunea **return**

5.6. Instrucţiunea **do while**

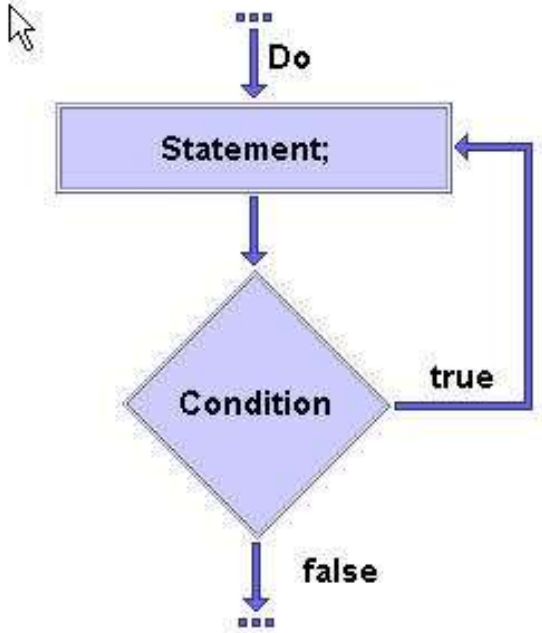
Forma instructiunii este:

```
do
    instructiune
while (expresie);
```

unde instrucţiune poate fi:

- ✓ instrucţiunea vidă
- ✓ instrucţiunea simplă
- ✓ instrucţiunea compusă

5.6. Instrucţiunea **do while**



Funcţionarea unei astfel de instrucţiuni se bazează pe executarea repetată a instrucţiunii *cât timp* condiţia este îndeplinită.

5.6. Instrucţiunea **do while**

Echivalenţa cu instrucţiunea *while*:

```
instructiune;
while(expresie)
        instructiune;
```

Exemplu de utilizare a instructiunii **do while**:

Să se scrie un program care tipăreşte numerele naturale de la 0 la 9 şi suma lor pe parcurs.

5.6. Instrucţiunea **do while**

La execuţia acestui program se obţine următorul rezultat:

```
numar=0 total=0
numar=1 total=1
numar=2 total=3
numar=3 total=6
numar=4 total=10
numar=5 total=15
numar=6 total=21
numar=7 total=28
numar=8 total=36
numar=9 total=45
```

```
#include<iostream.h>
int main(void)
{
    int numar = 0, total = 0;
    do{
            total = total + numar;
            cout<<"numar = "<<numar++<<"
    total = "<<total<<endl;
    }while(numar<10);
}
```

5. Instrucţiunile limbajului C++

5.1. Instrucţiunea **vidă**
5.2. Instrucţiunea **compusă**
5.3. Instrucţiunea **expresie**
5.4. Instrucţiunea **if**
5.5. Instrucţiunea **while**
5.6. Instrucţiunea **do while**
5.7. Instrucţiunea **for**
5.8. Instrucţiunea **switch**
5.9. Instrucţiunea **break**
5.10. Instrucţiunea **continue**
5.11. Instrucţiunea **goto**
5.12. Instrucţiunea **return**

5.7. Instrucţiunea **for**

Este una dintre cele mai “puternice” instrucţiuni ale limbajului C/C++, datorită formei sale.

Forma instrucțiunii este:

```
for(expresie1; expresie2; expresie3)
    instructiune;
```

expresie1 – reprezintă **secvenţa de iniţializarea a ciclului**

expresie2 – reprezintă **condiţia de terminare a ciclului**

expresie3 – reprezintă **secvenţa de reiniţializare a ciclului**

instrucţiune - **corpul ciclului**

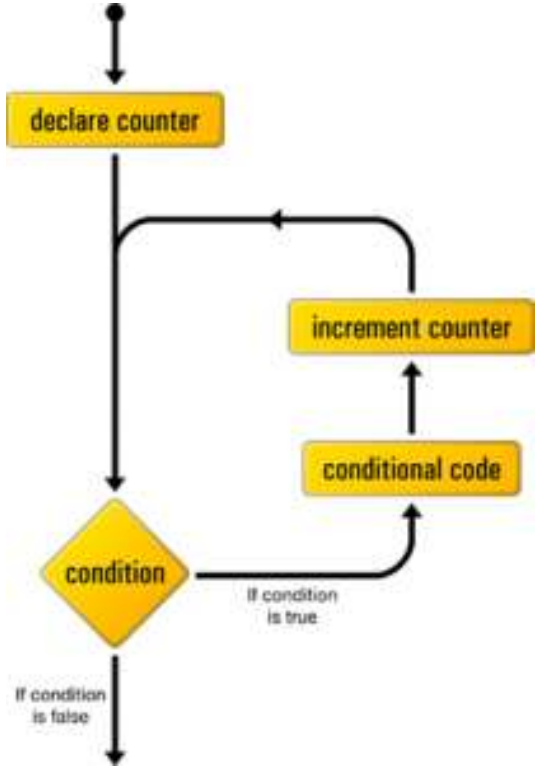
5.7. Instrucţiunea **for**

for(expresie1; expresie2; expresie3)
instructiune;

Funcţionarea unei astfel de instrucţiuni se bazează:

- ✓ pe executarea repetată a instrucţiunii,
- ✓ Verificarea expresiei2
- ✓ Executarea expresiei3

Cat timp expresia2 este îndeplinită.



5.7. Instrucţiunea **for**

Se stie că instrucţiunea ***for*** este de fapt o variantă particulară a instrucţiunii ***while***, drept pentru care se poate scrie echivalent astfel:

```
expresie1;
while(expresie2)
{
        instructiune;
        expresie3;
}
```

5.7. Instrucţiunea **for**

Invers, dacă avem:

while (expresie)
instructiune;

⇔

for(;expresie;)
instructiune;

5.7. Instrucţiunea **for**

Funcţionarea instrucţiunii ***for*** are loc astfel:

- Se porneşte ciclul repetitiv prin iniţializarea sa, adică prin execuţia **expresia1**
- iar apoi se evaluează **expresia2** şi dacă este adevărată se execută corpul ciclului, adică **instrucţiune**
- după aceea se execută **expresia3**, şi se reia evaluarea **expresiei2**, ş.a.m.d.

5.7. Instrucţiunea **for**

Observaţie:

expresia1, **expresia2**, **expresia3** pot să lipsească, dar este obligatorie prezenţa semnelor: **“;”**.

for(;;) ⇔ **while(1)** – buclă infinită

5.7. Instrucţiunea **for**

Exemplu: Acelaşi program de adunare a primelor n numere întregi, în varianta cu instrucţiunea ***for***, va avea o dimensiune mai mică:

```
#include<iostream.h>
int main(void)
{
   int i, n, s = 0;
   cout<<"Dati numarul n =";
   cin>>n;
   for(i=1; i<=n; i++)
         s = s + i;
   cout<<"Suma primelor"<<n<<"
   numere intregi este "<<s;
}
```

Exemplu de program C++ - instructiunea for

Enunţ:

Fie un număr natural n. Să se scrie un algoritm care să calculeze factorialul numărului dat.

(factorial = *produsul numerelor naturale mai mici sau egale decat n*)

Exemplu:

Pentru n = 5, se va afişa valoarea

p = 1*2*3*4*5=120.

Exemplu de program C++ - instructiunea for

Pas 1: Stabilim care sunt datele de intrare, adică cele care vor fi prelucrate cu ajutorul algoritmului, împreună cu datele de ieşire.

În cazul problemei date, avem:

Date de intrare: n = număr natural

Date de ieşire: factorialul numărului dat = p

Exemplu de program C++ - instructiunea for

Pas 2: Analiza problemei

La începutul problemei, *vom iniţializa valoarea produsului numerelor cu 1*.

Apoi, *într-un ciclu repetitiv* vom calcula produsul numerelor naturale aflate intre 1 si n.

Exemplu de program C++ - instructiunea for

Pas 3:

Scrierea algoritmului în pseudocod:

```
natural n, p, i
citeşte n
p <- 1
pentru i=1,n execută
        p <- p * i
sfârşit pentru
scrie p
stop
```

Exemplu de program C++ - instructiunea for

Pas 4: Implementarea algoritmului în limbajul de programare C++:

```
#include<iostream.h>
int main(void)
{
   int n, p, i;
   cin>>n;
   p = 1;
   for(i = 1; i<=n; i++)
        p = p * i;
   cout<<p;
}
```

Exemplu de program C++ - instructiunea for

Pas 5: Testarea algoritmului pe date de intrare diferite şi verificarea rezultatelor.

Exemplul 1:

Pentru valoarea n=5 obţinem următorul rezultat:

120

Exemplul 2:

Pentru valoarea n=7 obţinem următorul rezultat:

5040

Exemplu de program C++ - instructiunea do while

Enunț:

Fie un număr natural n de cinci cifre. Să se scrie un algoritm care să calculeze suma cifrelor numărului dat.

Exemplu:

Pentru n = 2178, se va afişa valoarea s = 2+1+7+8 = 18

Exemplu de program C++ - instructiunea do while

Pas 1: Stabilim care sunt datele de intrare, adică cele care vor fi prelucrate cu ajutorul algoritmului, împreună cu datele de ieşire.

În cazul problemei date, avem:

Date de intrare: n număr natural

Date de ieşire: suma cifrelor = s.

Exemplu de program C++ - instructiunea do while

Pas 2: Analiza problemei

La începutul problemei, vom iniţializa valoarea sumei cifrelor numărului n dat cu 0.

Apoi, *într-un ciclu repetitiv* vom calcula suma cifrelor numărului, ştiind că:

- o cifră a unui număr scris în baza 10 este dată de **restul împărțirii** la 10 - **n%10**,
- iar numărul fără ultima cifră este dat de **câtul împărțirii** la 10 - **n/10**.

Exemplu de program C++ - instructiunea do while

Pas 3:

Scrierea algoritmului în pseudocod:

```
natural n, s
citeşte n
s <- 0
repetă
        s <-  s + n % 10
        n <- n / 10
până când n = 0
scrie s
stop
```

Exemplu de program C++ - instructiunea do while

Pas 4: Implementarea algoritmului în limbajul de programare C++:

```
#include<iostream.h>
int main(void)
{
   int n, s;
   cin>>n;
   s = 0;
   do
   {
        s = s + n  % 10;
        n = n / 10;
   }while( n != 0 );
   cout<<s;
}
```

Exemplu de program C++ - instructiunea do while

Pas 5: Testarea algoritmului pe date de intrare diferite şi verificarea rezultatelor.

Exemplul 1:

Pentru valoarea n=123 obţinem următorul rezultat:

6

Exemplul 2:

Pentru valoarea n=5378 obţinem următorul rezultat:

23

Exemplu de program C++ - instructiunea do while

Enunţ:

Să se scrie un program care generează toate numerele perfecte până la o valoare dată, n.

Un număr perfect este *egal cu suma divizorilor lui, inclusiv 1* (exemplu: 6=1+2+3).

Exemplu:

Pentru n = 1000, se vor afişa valorile 6, 28, 496

Exemplu de program C++ - instructiunea do while

Pas 1: Stabilim care sunt datele de intrare, adică cele care vor fi prelucrate cu ajutorul algoritmului, împreună cu datele de ieşire.

În cazul problemei date, avem:

Date de intrare: n număr natural

Date de ieşire: numerele perfecte mai mici sau egale decât n.

Exemplu de program C++ - instructiunea do while

Pas 2: Analiza problemei

La începutul problemei, *vom lua toate valorile de la 1 la n*, şi *pentru fiecare valoare i o vom verifica dacă este sau nu număr perfect*.

Numerele perfecte obţinute le vom afişa.

Exemplu de program C++ - instructiunea do while

Pas 3:

Scrierea algoritmului în pseudocod:

```
natural n, i, j, s
citeşte n
i <- 1
repetă
        s <- 0
        j <- 1
        repetă
                dacă i % j = 0 atunci
                        s <- s + j
                sfârşit dacă
                j <- j + 1
        până când j > i/2
        dacă s = i atunci
                scrie i
        sfârşit dacă
        i <- i + 1
până când i > n
stop
```

Exemplu de program C++ - instructiunea do while

Pas 4: Implementarea algoritmului în limbajul de programare C++:

```
#include<iostream.h>
int main(void)
{
    int n, i, j, s;
    cin>>n;
    i = 1;
    do{
        s = 0;
        j = 1;
        do{
            if( i % j == 0 )
                s = s + j;
            j = j + 1;
        }while(j <= i/2);
        if( s == i )
            cout<<i<<" ";
        i = i + 1;
    }while(i <= n);
}
```

Exemplu de program C++ - instructiunea do while

Pas 5: Testarea algoritmului pe date de intrare diferite şi verificarea rezultatelor.

Exemplul 1:

Pentru valoarea n=100 obţinem următoarele rezultate:

6 28

Exemplul 2:

Pentru valoarea n=10000 obţinem următoarele rezultate:

6 28 496 8128

Exemplu de program C++ - instructiunea for

Enunț:

Să se scrie un program care generează toate numerele prime până la o valoare dată, n.

Un număr x este prim dacă are ca divizori doar valorile 1 şi x.

Exemplu:

Pentru n = 22, se vor afişa valorile:

2,3,5,7,11,13,17,19

Exemplu de program C++ - instructiunea for

Pas 1: Stabilim care sunt datele de intrare, adică cele care vor fi prelucrate cu ajutorul algoritmului, împreună cu datele de ieşire.

În cazul problemei date, avem:

Date de intrare: n număr natural

Date de ieşire: numerele prime mai mic decat n

Exemplu de program C++ - instructiunea for

Pas 2: Analiza problemei

Intr-un ciclu repetitiv de la 1 la n vom verifica toate valorile daca respecta proprietatea de numar prim.

Vom presupune, la începutul problemei, că numărul i dat este prim, şi vom specifica acest lucru cu ajutorul unei variabile de tip întreg, căreia îi vom da valoarea **1**.

- *Apoi vom evalua, pe rând, toate valorile începând cu valoarea 2 şi până la i/2, ca să determinăm dacă sunt divizori ai numărului i dat.*
- *Dacă găsim un singur divizor printre aceste numere, atunci vom acorda valoarea **0** variabilei de tip întreg de la începutul verificarii conditiei de numar prim.*
- *La sfârşit vom verifica care este valoarea variabilei de tip întreg şi vom afişa numarul i.*

Exemplu de program C++ - instructiunea for

Pas 3:

Scrierea algoritmului în pseudocod:

```
natural n,i,j
logic p
citeşte n
pentru i = 2, n execută
  p <-  adevărat
  pentru j = 2, i/2 execută
     dacă i % j = 0 atunci
        p <- fals
     sfârşit dacă
  sfârşit pentru
  dacă p = adevărat atunci
        scrie i,` ’
  sfârşit dacă
sfârşit pentru
stop
```

Exemplu de program C++ - instructiunea for

Pas 4: Implementarea algoritmului în limbajul de programare C++:

```
#include<iostream.h>
int main(void)
{
    int n, prim, i, j;
    cin>>n;
    for(i = 2; i<=n;  i++)
    {
        prim = 1;
        for(j = 2; j<=i/2;  j++)
            if(i % j == 0 ) prim = 0;
        if( prim == 1)
            cout<<i<<" ";
    }
}
```

Exemplu de program C++ - instructiunea for

Pas 5: Testarea algoritmului pe date de intrare diferite şi verificarea rezultatelor.

Exemplul 1:

Pentru valoarea n=53 obţinem rezultatele:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53

Exemplul 2:

Pentru valoarea n=12 obţinem următorul rezultat:

2 3 5 7 11

5. Instrucţiunile limbajului C++

5.1. Instrucţiunea **vidă**
5.2. Instrucţiunea **compusă**
5.3. Instrucţiunea **expresie**
5.4. Instrucţiunea **if**
5.5. Instrucţiunea **while**
5.6. Instrucţiunea **do while**
5.7. Instrucţiunea **for**
5.8. Instrucţiunea **switch**
5.9. Instrucţiunea **break**
5.10. Instrucţiunea **continue**
5.11. Instrucţiunea **goto**
5.12. Instrucţiunea **return**

5.8. Instrucţiunea **switch**

Instrucţiunea ***switch*** funcţionează astfel:

Se evaluează expresia şi în funcţie de rezultat se compară cu c_1, c_2, . . ., c_n şi când expresia este egală cu c_1 atunci se execută şirul de instrucţiuni corespunzător, şi cu instrucţiunea **break** se sare la sfârşitul instrucţiunii ***switch***, la fel se întâmplă şi dacă expresia este egală cu c_2 ,sau cu c_3 , sau cu c_n.

```
switch (expresie)
{
    case c1 :
            sir_instructiuni_1;
              break;
    case c2 :
            sir_instructiuni_2;
              break;
          . . . . . .
    case cn :
           sir_instructiuni_ n;
              break;
    default: sir_instructiuni;
}
```

5.8. Instrucţiunea **switch**

Instrucţiunea ***switch*** este o instrucţiune de tip decizie multiplă astfel încât se poate scrie echivalent folosind instrucţiunea de decizie simplă ***if***:

```
if (expresie==c1)
          sir_instructiuni_1;
  else
      if (expresie==c2)
          sir_instructiuni_2;
       . . . . . .
          else
              if (expresie==cn)
                  sir_instructiuni_ n;
              else sir_instructiuni;
```

5.8. Instrucţiunea **switch**

Prezentăm în continuare *un program care numără vocalele şi consoanele din alfabet.*

De observat că unele din instrucţiunile ***case*** se execută în cascadă, pentru calculul vocalelor, iar pentru consoane se foloseşte cazul ***default***:

```
#include<iostream.h>
int main(void)
{
    char litera;
    int nr_vocale = 0, nr_consoane = 0;
    for (litera = 'A'; litera <= 'Z'; litera++)
        switch (litera) {
                case 'A':
                case 'E':
                case 'I':
                case 'O':
                case 'U': nr_vocale++;
                break;
                default: nr_consoane++;
        }
    cout<<"\nNumarul de vocale este "<<nr_vocale;
    cout<<"\nNumarul de consoane este "<< nr_consoane;
}
```

5. Instrucţiunile limbajului C++

5.1. Instrucţiunea **vidă**
5.2. Instrucţiunea **compusă**
5.3. Instrucţiunea **expresie**
5.4. Instrucţiunea **if**
5.5. Instrucţiunea **while**
5.6. Instrucţiunea **do while**
5.7. Instrucţiunea **for**
5.8. Instrucţiunea **switch**
5.9. Instrucţiunea **break**
5.10. Instrucţiunea **continue**
5.11. Instrucţiunea **goto**
5.12. Instrucţiunea **return**

5.9. Instrucţiunea **break**

```
break;
```

Instrucţiunea întrerupe execuţia instrucţiunilor ***while***, ***do while***, ***for*** şi ***switch***, determinând astfel ieşirea forţată dintr-un ciclu repetitv.

Exemplu:

```
for(;;)
{
    . . . . .
    break;
}
```

5.9. Instrucţiunea **break**

Exemplu:

Prezentăm în continuare, un program care folosind instrucţiunea ***break***, *afisează numerele întregi aflate între 1 şi 100 şi apoi de la 100 la 1*.

De fiecare dată când număr ajunge la valoarea 50, instrucţiunea ***break*** face ca execuţia ciclului să se oprească:

```
#include<iostream.h>
int main(void)
{
    int numar;
    for(numar = 1; numar<=100; numar++)
    {
        if(numar == 50) break;
        cout<<"  "<<numar;
    }
    cout<<"\nCel de-al doilea ciclu
    repetitiv";
    for(numar = 100; numar>=1; numar--)
    {
        if(numar == 50) break;
        cout<<"  "<<numar;
    }
}
```

5. Instrucţiunile limbajului C++

5.1. Instrucţiunea **vidă**
5.2. Instrucţiunea **compusă**
5.3. Instrucţiunea **expresie**
5.4. Instrucţiunea **if**
5.5. Instrucţiunea **while**
5.6. Instrucţiunea **do while**
5.7. Instrucţiunea **for**
5.8. Instrucţiunea **switch**
5.9. Instrucţiunea **break**
5.10. Instrucţiunea **continue**
5.11. Instrucţiunea **goto**
5.12. Instrucţiunea **return**

5.10. Instrucţiunea **continue**

Se referă la instrucţiunile de ciclare: ***for***, ***while*** şi ***do while***.

La întâlnirea ei ciclurile ***while*** şi ***do while*** *se continuă cu reevaluarea condiţiei de ciclare* iar în ciclul ***for*** se continuă cu *secvenţa de reiniţializare a ciclului şi apoi cu reevaluarea ciclului*.

5.10. Instrucţiunea **continue**

Exemplu:
Prezentăm în continuare, un program care folosind instrucţiunea *continue* într-un ciclu *for* şi într-un ciclu *while*, *afişează numerele pare şi impare aflate între 1 şi 100*:

```
#include<iostream.h>
int main(void)
{
    int numar;
    cout<<"Numerele pare dintre 1 si 100 sunt: ";
    for (numar = 1; numar <= 100; numar++)
    {
        if(numar % 2 != 0) continue;
        cout<<"  "<<numar;
    }
    cout<<"\nNumerele impare dintre 1 si 100 sunt: ";
    numar=0;
    while(numar <= 100)
    {
        numar++;
        if(numar % 2==0) continue;
        cout<<"  "<<numar;
    }
}
```

5. Instrucţiunile limbajului C++

5.1. Instrucţiunea **vidă**
5.2. Instrucţiunea **compusă**
5.3. Instrucţiunea **expresie**
5.4. Instrucţiunea **if**
5.5. Instrucţiunea **while**
5.6. Instrucţiunea **do while**
5.7. Instrucţiunea **for**
5.8. Instrucţiunea **switch**
5.9. Instrucţiunea **break**
5.10. Instrucţiunea **continue**
5.11. Instrucţiunea **goto**
5.12. Instrucţiunea **return**

5.11. Instrucţiunea **goto**

goto eticheta;

Este instrucţiunea pentru salt necondiţionat.

unde eticheta este un nume care prefixează o instrucţiune.

Exemplu: Prezentăm în continuare, un program care folosind instrucţiunea ***goto***, afişează numerele întregi aflate între 1 şi 100:

```
int main(void)
{
   int numar=1;
   eticheta: cout<<"  ", numar++;
   if (numar <= 100) goto eticheta;
}
```

5. Instrucţiunile limbajului C++

5.1. Instrucţiunea **vidă**
5.2. Instrucţiunea **compusă**
5.3. Instrucţiunea **expresie**
5.4. Instrucţiunea **if**
5.5. Instrucţiunea **while**
5.6. Instrucţiunea **do while**
5.7. Instrucţiunea **for**
5.8. Instrucţiunea **switch**
5.9. Instrucţiunea **break**
5.10. Instrucţiunea **continue**
5.11. Instrucţiunea **goto**
5.12. Instrucţiunea **return**

5.12. Instrucţiunea **return**

Instrucţiunea return

1. return;
2. return expresie;
3. return (expresie);

Se foloseşte în funcţii atunci când:

- *se întoarce în funcţia apelantă o valoare* (formele 2 şi 3)
- sau *într-o funcţie care nu întoarce nici o valoare* (funcţioneaza ca o procedură) – forma 1.

Probleme propuse spre rezolvare:

1) Să se afişeze toate numerele palindrom mai mari decat 10 şi mai mici decat un număr dat, n.

Exemplu:

Date de intrare: n=110
Date de ieşire:
11 22 33 44 55 66 77 88 99 101

Probleme propuse spre rezolvare:

2) Să se determine toate tripletele de numere a, b, c cu proprietăţile: 1<a<b<c<100; a+b+c se divide cu 10.

Exemplu:

Date de intrare: -
Date de ieşire: 95 96 99 si 95 97 99 sunt ultimele doua triplete

Probleme propuse spre rezolvare:

3) Să se afişeze toate numerele de două cifre care adunate cu răsturnatul lor dau 55.

Exemplu:

Date de intrare: -
Date de ieşire: 14 41; 23 32; 32 23; 41 14

Pentru alte informații teoretice și aplicative legate de acest capitol se recomandă următoarele referințe bibliografice:

1. Adrian Runceanu, Mihaela Runceanu, ***Noțiuni de programare în limbajul C++***, Editura Academica Brâncuși, Târgu-Jiu, 2012 (www.utgjiu.ro/editura)

2. Adrian Runceanu, **Programarea şi utilizarea calculatoarelor**, Editura Academica Brâncuși, Târgu-Jiu, 2003 (www.utgjiu.ro/editura)

3. Octavian Dogaru, **C++ - teorie şi practică**, volumul I, Editura Mirton, Timişoara, 2004 (www.utgjiu.ro/editura)

Recapitulare pseudocod – limbaj C++

	Pseudocod	C++
Tipuri de date simple:	natural intreg real logic	**unsigned** **int, long** **float, double** **bool**
CITIRE date	citeste v1,v2	**cin>>v1>>v2 ;**
AFISARE date	scrie expresie1, expresie2	**cout<<expresie1<<expresie2**
ATRIBUIRE	v←expresie	**v=expresie;** **v1=v2=...=v3=expresie;**
STRUCTURA DECIZIONALA	daca conditie atunci instructiune 1 altfel instructiune 2	**if (conditie) instructiune 1;** **else instructiune 2;**
STRUCTURA executa... cat timp	executa instructiune cat timp conditie	**do** **instructiune;** **while (conditie);**
PENTRU	pentru v←vali, valf, pas executa instructiune	**for(v=vali ; v<=valf ; v=v+pas)** **instructiune ;**
INSTRUCTIUNE COMPUSA	**| instructiune 1;** **| instructiune 2;** **| .....** **| instructiune n;** **└**	**{** **instructiune 1;** **instructiune 2;** **.....** **instructiune n;** **}**

Recapitulare elemente de limbaj C++

Structura unui program C++

Biblioteci	#include <iostream.h>
	int main()
	{
	.....
	instructiuni
	}

Antet functie main { declaratii de: tipuri, constante, variabile Citire date intrare Prelucrare date Afisare }	

Sfârşit capitol!