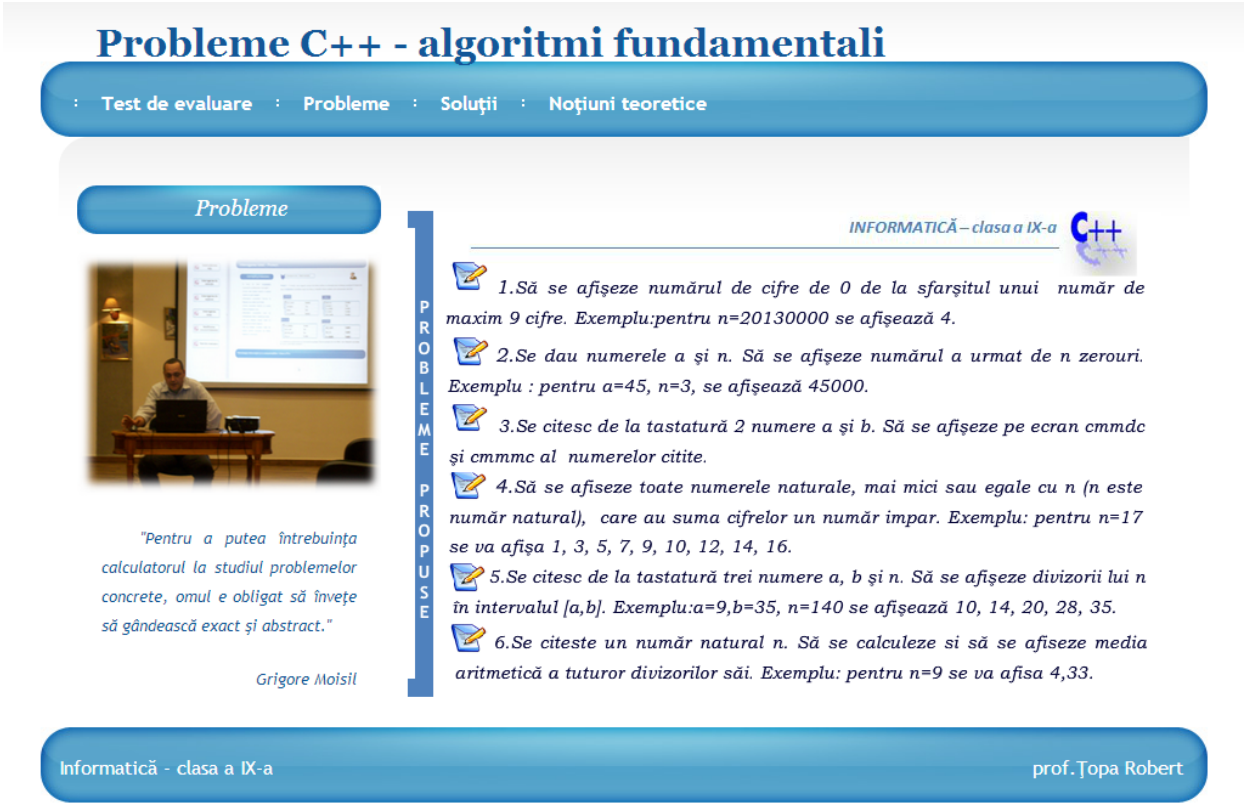
NOŢIUNI TEORETICE ŞI PROBLEME
OPERATORI, STRUCTURI ALTERNATIVE ŞI REPETITIVE, ALGORITMI FUNDAMENTALI



"Fiecare greşeală este o ocazie de a învăța. Totul este să nu comiți aceeaşi greşeală în mod repetat - ar fi o prostie. Însă comite cât mai multe greşeli noi de care eşti în stare; nu trebuie să-ti fie teamă, căci acesta este singurul mod în care natura îți permite să înveți."

OSHO

Cuprins

NOȚIUNI TEORETICE DESPRE LIMBAJUL C++ - PREZENTARE GENERALĂ

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:

- o etapă în care este implicat omul şi anume cea de trecere de la problema reală la transpunerea într-un limbaj de programare.
- 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**.

Atunci când scriem un program într-un anumit limbaj de programare trebuie să luăm în considerare următoarele:

- declararea, scrierea și citirea setului de **date de intrare** (cele care trebuie să fie prelucrate);
- să execute asupra lor suita standard de operaţiuni;
- şi să livreze **datele de ieşire** (adică rezultatele).

Un ***program*** scris în limbajul C++ este compus din unul sau mai multe ***fişiere sursă***. Un fişier sursă este un fişier text care conţine codul sursă al unui program. Fiecare fişier sursă conţine una sau mai multe ***funcţii*** şi eventual, referinţe către unul sau mai ***fişiere header***. Funcţia principală a unui program este numită ***main***.

Exemplu

```
//fisierul header iostream pentru operatiile de scriere si citire
#include <iostream>
/*fisierul header cmath folosit pentru a apela
  functii matematice */
#include <cmath>
using namespace std;
//functia principala main care poate apela alte functii
int main()
{
    //partea declarativa-variabila a de tip intreg
    int a;
    //si variabila b de tip real
    float b;
    //scrierea lui a - apare pe ecran a=
    cout<<"a=";
    //citirea lui a de la tastatura
    cin>>a;
    //instructiune
    b=sqrt(a);
    //afisarea pe ecran a valorii variabilei b
    cout<<"Valoarea lui b este "<<b<<".";
}
```

ELEMENTELE DE BAZĂ ALE LIMBAJULUI C++

Vocabularul limbajului

a)Setul de caractere

Setul de caractere reprezintă ansamblul de caractere cu ajutorul cărora se poate realiza un program C++. Acesta este alcătuit din:

- litere mari și mici ale alfabetului englez(A-Z,a-z);
- cifrele sistemului de numerație în baza 10(0-9);
- caractere speciale(+,-,/,=,%,<,>,:,;,#,$,@,blank(spațiu)).

b)Identificatori. Cuvinte cheie

Identificatorul este o succesiune de litere, cifre sau caracterul special underscor(_) din care prima nu trebuie să fie cifră. Cu ajutorul identificatorilor se asociază nume constantelor, variabilelor, funcțiilor, etc. Exemple de identificatori: **a, c1, contor_cifre**, etc.

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.

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**
bool	**catch**	**class**	**delete**	**namespace**	**inline**	**new**	**operator**
private	**this**	**using**	**template**	**virtual**	**mutable**	**cin**	**cout**

c)Comentarii

Un comentariu, în limbajul C++, începe cu semnul //comentariu - pentru a scrie un comentariu pe o singură linie – sau cu semnul /* comentariu */ - pentru a scrie un comentariu pe mai multe linii. Comentariile nu au niciun efect asupra comportamentului programului. Programatorul poate să le folosească pentru a include explicaţii scurte sau observaţii asupra codului sursă.

Scrierea și citirea în limbajul C++

Pentru a realiza scrieri pe ecran se folosește fluxul **cout<<**. Cuvântul cheie **cout** este acronimul de la **console output** și se mai numește - stream de ieșire. Forma generală a acestui stream este **cout<<a<<b<<c<<……<<n;**

Pentru a realiza citiri de la tastatură se folosește fluxul **cin>>**. Cuvântul cheie **cin** este acronimul de la **console input** și se mai numește stream de intrare. Forma generală a acestui stream este **cin>>a>>b>>……>>n;**

Aplicație

Click aici Exemplu și identifică în program operațiile de scriere și citire. Folosind un mediu de dezvoltare al programelor scrise cu ajutorul limbajului C++(Code Blocks, MinGW, Borland C++, Dev C++) sau folosind compilatorul online de aici, scrie și urmărește execuția programului de mai jos.

```
#include <iostream>
using namespace std;
int main()
{
    int x,y,z;
    cout<<"Introduceti valoarea lui x:";cin>>x;
    cout<<"Introduceti valoarea lui y:";cin>>y;
    z=x+y;
    cout<<"Suma celor doua numere este "<<z<<endl;
    z=x*y;
    cout<<"Produsul celor doua numere este "<<z<<endl;
    z=x/y;
    cout<<"Catul impartirii celor doua numere este "<<z<<endl;
    z=x%y;
    cout<<"Restul impartirii celor doua numere este "<<z;
}
```

Tipuri de date, constante, variabile

a)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.

TIP	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: 7.8965, -4.123, 7.0, 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.

Modificatorii de tip. Limbajul C++ oferă pe lîngă cele 5 tipuri de bază prezentate mai sus, un set de modificatori de tip:***unsigned***(fără semn), ***long***(lung), ***signed***(cu semn), ***register***(registru), ***short***(scurt). 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 faţa tipului respectiv.*

b)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:

constantă întreagă - se reprezintă sub forma unei înşiruiri de cifre: 6,456,1234.

constantă flotantă - 6.023e-23 = $6.023*10^{-23}$

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')

constanta şir sau şir de caractere - acest tip de constantă apare ca o succesiune de caractere scrise între ghilimele. Exemplu:"program","CalculaTor".

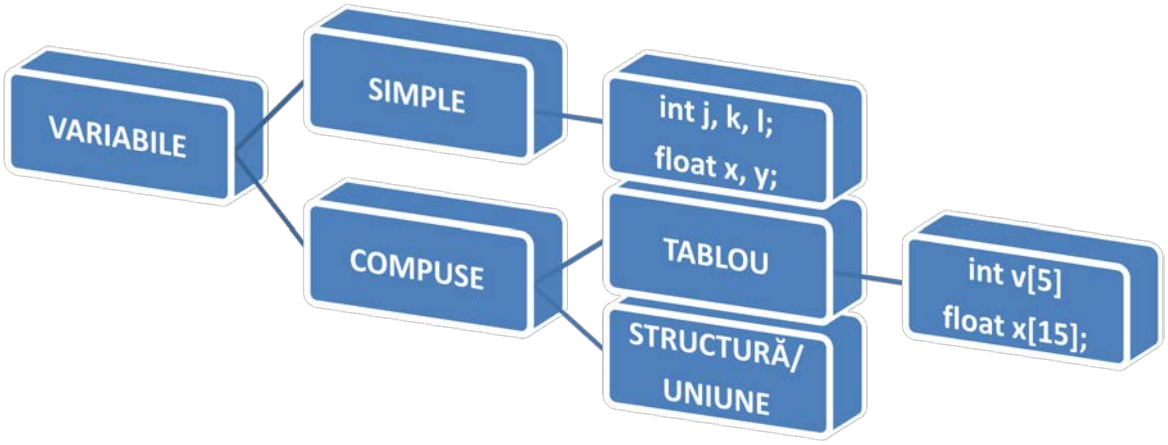
c)Variabile

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

Numim **variabilă** *o denumire (identificator) pe care compilatorul o asociază cu o anumită zonă de memorie*. Când se declară o variabilă, trebuie specificat numele ei cât şi tipul de date asociat. Exemple:

int a	**variabila a este de tip întreg**
float b	variabila b este de tip real
char d	variabila d este de tip caracter
void	variabila fără tip

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. Variabilele pot fi:



Expresii. Operatori C++

Expresia este alcătuită dintr-unul sau mai mulţi **operanzi** legaţi între ei prin **operatori** pentru a efectua anumite operații (calcule, atribuiri, apelări de funcții, etc.). Limbajul C++ conține un set puternic de operatori. Cei mai importanți operatori ai limbajului C++ sunt:

- Operatorii aritmetici
- Operatorii relaționali
- Operatorii de incrementare și decrementare
  a) prefixați
  b) postfixați
- Operatorii logici
- Operatorii de asignare(atribuire)
- Operatorul conditional
- Operatorul de conversie explicită
- Operatorul dimensiune sizeof

a)Operatori aritmetici

Nr. crt.	Operator	Semnificaţie	Tipul datelor	Exemplu
1.	+	adunare	numeric	z = y + x;
2.	-	scădere	numeric	z = y – x;
3.	*	înmulţire	numeric	z = y * x;
4.	/	împărţire	numeric	z = y / x;
5.	%	modulo	întreg	z = y % x;

Operatorii +, -, /, * pot fi utilizaţi cu orice tip de dată, pe când operatorul **modulo** poate fi folosit doar cu datele de tip întreg. Operatorul **modulo** reţine restul unei împărţiri întregi.

Aplicație

Folosind un mediu de dezvoltare al programelor scrise cu ajutorul limbajului C++(Code Blocks, MinGW, Borland C++, Dev C++) sau folosind compilatorul online de aici, scrie și urmărește execuția programului de mai jos:

```
#include <iostream>
using namespace std;
int main()
{
    int a,b=8;
    float x,y;
    char c='a',d='b';
    a=5;
    x=3.14;y=2;
    cout<<a+b<<endl;
    cout<<a-x<<endl;
    cout<<b-y<<endl;
    cout<<a/x<<endl;
    cout<<c<<d<<endl;
    cout<<c+d<<endl;
    cout<<c-d<<endl;
    cout<<b%a<<endl;
    cout<<a+c<<endl;
    cout<<x-d;
}
```

b)Operatori relaționali

Nr. crt.	Operator	Operaţie
1.	<	mai mic
2.	>	mai mare
3.	< =	mai mic sau egal
4.	> =	mai mare sau egal
5.	==	egal
6.	!=	diferit

Rezultatul unui operator relaţional nu poate fi decât **true** (sau 1), respectiv **false** (sau 0).

c)Operatori de incrementare și decrementare

Operatorul de incrementare (++) şi **decrementare** (--) măreşte sau micşorează valoarea operandului său cu 1 (unu). Operatorii de incrementare şi decrementare se pot aplica variabilelor, dar nu şi constantelor, aceştia putând fi prefixaţi sau postfixaţi.

Prefixaţi	Postfixaţi
++ operand	operand --
-- operand	operand ++

Diferenţa dintre operatorii **postfixaţi** şi operatorii **prefixaţi** este aceea că dacă operandul este **postfixat** atunci mai întâi se foloseşte valoarea lui nemodificată, iar apoi se aplică operatorul respectiv. Dacă operandul este **prefixat** atunci mai întâi se aplică operatorul respectiv şi abia apoi se foloseşte valoarea lui.

Aplicație. Folosind un mediu de dezvoltare al programelor scrise cu ajutorul limbajului C++(Code Blocks, MinGW, Borland C++, Dev C++) sau folosind compilatorul online de aici, scrie și urmărește execuția programului de mai jos:

```
#include <iostream>
using namespace std;
int main()
{
    int a=5,b=7;
    cout<<(a>b)<<endl;
    cout<<(a<b)<<endl;
    cout<<(a!=b)<<endl;
    cout<<(a==b)<<endl;
    cout<<a++<<endl;
    cout<<a<<endl;
    a++;b--;
    cout<<a<<endl;
    cout<<b<<endl;
    cout<<++b<<endl;
    cout<<b++<<endl;
    cout<<a--+b<<endl;
    cout<<--a+b<<endl;
}
```

d)Operatorii logici

Nr. crt.	Operator	Operaţie
1.	!	negarea logică
2.	&&	ŞI logic
3.	||	SAU logic

Rezultatul unui operator logic nu poate fi decât **true** (sau 1), respectiv **false** (sau 0). Tabelul de adevăr este prezentat mai jos:

Nr. crt.	A	B	A && B	A || B	! A
1.	0	0	0	0	1
2.	0	1	0	1	1
3.	1	0	0	1	0
4.	1	1	1	1	0

e)Operatorii de asignare(atribuire)

Nr. crt.	Operator	Forma lungă	Forma scurtă
1.	=	x=y	
2.	+=	x = x + y	x + = y
3.	-=	x = x - y	x - = y
4.	*=	x = x * y	x * = y
5.	/=	x = x / y	x / = y
6.	%=	x = x % y	x % = y

Aplicație. Folosind un mediu de dezvoltare al programelor scrise cu ajutorul limbajului C++(Code Blocks, MinGW, Borland C++, Dev C++) sau folosind compilatorul online de aici, scrie și urmărește execuția programului de mai jos:

```
#include <iostream>
using namespace std;
int main()
{
    int a=10,b=15;
    cout<<((a>b)&&(a==b))<<endl;
    cout<<((a<=b)||(a>=b))<<endl;
    cout<<((a!=b)&&(a==(b-5)||(b==b)))<<endl;
   a+=b;
   cout<<a<<endl;
   b*=a;
   cout<<b<<endl;
   b/=a;
   cout<<b<<endl;
   a%=b;
   cout<<a;
}
```

f)Operatorul condiţional

Operatorii condiţionali se utilizează în evaluarea expresiilor care prezintă alternative. Forma generală este următoarea:

exp1 ? exp2 : exp3;

unde *exp1*, *exp2*, *exp3* sunt expresii. Dacă valoarea expresiei *exp1* este adevărată atunci valoarea şi tipul expresiei va fi *exp2*; altfel valoarea şi tipul va fi *exp3*.
? : sunt **operatorii condiţionali** şi trebuie să fie folosiţi împreună.
Exemplu: a=5
b = a > 7 ? 15 : 20;
În acest exemplu variabilei *b* i se atribuie valoarea *20*. Dacă *a* ar fi fost mai mic decât *5*, lui *b* i s-ar fi atribuit valoarea *15*.

g)Operatorul de conversie explicită

Acest operator realizează o schimbare temporară a tipului unei expresii şi are următoarea formă generală:

(tip)expresie;

unde: *tip* este unul dintre tipurile de date admise în limbajul C++(int, float, char, etc.).

h)Operatorul dimensiune sizeof

Operatorul dimensiune are următoarea formă generală:

sizeof (data)

Acesta returnează lungimea în octeţi a unei date. *Data* poate fi numele unei variabile simple, al unui tablou, al unei structuri, al unui tip sau referirea la elementul unui tablou sau structură.
Aplicație. Folosind un mediu de dezvoltare al programelor scrise cu ajutorul limbajului C++(Code Blocks, MinGW, Borland C++, Dev C++) sau folosind compilatorul online de aici, scrie și urmărește execuția programului de mai jos:

```
#include <iostream>
using namespace std;
int main()
{
    int a=10,b=15;
    char c='A';
    cout<<(a>b?"DA":"NU")<<endl;
    cout<<a/b<<endl;
    cout<<(float)a/b<<endl;
    cout<<c<<endl;
    cout<<(int)c<<endl;

cout<<sizeof(c)<<"octet"<<sizeof(a)<<"octeti"<<sizeof(b)<<"octeti ";
}
```

FIŞĂ DE LUCRU - OPERATORI C++

1.Se citeşte un număr natural care reprezintă timpul exprimat în minute. Scrieţi programul care afişează timpul exprimat în ore şi secunde.

2.Scrieţi un program care să testeze un caracter introdus de la tastatură. Dacă este literă mare, să afişeze mesajul "Literă mare", dacă este literă mică să se afişeze mesajul "Literă mică", altfel să se afişeze mesajul "Nu este literă".

3.Scrieţi un program care citeşte de la tastatură un număr natural cu trei cifre şi care afişează apoi numărul obţinut prin eliminarea cifrei din mijloc.

4. Scrieţi un program care citeşte de la tastatură un număr natural cu patru cifre şi care afişează pe câte un rând cifrele numărului.

5. Scrieţi un program care citeşte de la tastatură un număr natural cu patru cifre şi care afişează numărul obţinut prin eliminarea cifrei sutelor. Modificaţi programul, astfel încât, să afişeze numărul obţinut prin eliminarea cifrei zecilor.

6.Se citesc de la tastatură trei numere a, b şi c. Să se afişeze valoarea maximă. Modificaţi programul, astfel încât, să afişeze valoarea minimă.

7.Se citeşte de la tastatură un număr n. Să se verifice dacă este un număr pozitiv sau negativ.

8. Se citesc de la tastatură 3 numere a, b şi c care reprezintă laturile unui triunghi oarecare. Să se calculeze aria triunghiului.

9. Indicaţi rezultatele pe care le afişează programul următor. Explicaţi obţinerea acestor rezultate.

```
{
int x=2, y=7, z,u;
u=x*(y-2)%3;
      cout<<"u="<<u<<endl;
z=u+x;
x=x*y;
      cout<<"x="<<x<<" y="<<y<<" z="<<z<<" u="<<u<<endl;
      x=-y*z%3+u;
      cout<<"x="<<x<<endl;
      z=(x-y)*(u-x);
    cout<<"z="<<z<<endl;
}
```

10. Scrieţi un program care converteşte gradele Celsius în grade Fahrenheit conform formulei f=(9/5)*c+32. Datele se citesc de la tastatură.

11. Se introduc 2 numere, a şi b şi un număr k. Să se afişeze un mesaj dacă fracţia a/b poate fi simplificată prin k.

12. Se introduc 2 numere. Să se afişeze un mesaj dacă aceste numere sunt consecutive.

13. Fie variabilele x,y şi u de tipul int. Scrieţi o instrucţiune care măreşte valoarea variabilei u cu câtul întreg al împarţirii lui x la y ?

14. Scrieţi 3 valori ce pot fi citite pentru variabila y astfel încât programul de mai jos să tipărească 1 ?

```
{int x=2, y, z;
cin>>y;
x++;
z=y+3*x;
cout<<((z%2==0 && x>=1) ? 1 : 0 );
}
```

15. Ce se afişează în urma execuţiei secvenţei de instrucţiuni de mai jos, dacă pentru n se citeşte valoarea 815?

```
{
int n, a, b, c, x, w, q;
cout<<"n="; cin>>n;
a=n/100;
b=n/10%10;
c=n%10;
(a>b ? x=a, a=b, b=x : x);
(b>c ? x=b, b=c, c=x : x);
(a>b ? x=a, a=b, b=x : x);
cout<<a<<" "<<b<<" "<<c<<endl;
w=a*100+b*10+c;
q=c*100+b*10+a;
cout<<w<<"  "<<q;
}
```

FIŞĂ DE LUCRU – STRUCTURA ALTERNATIVĂ

Structura alternativă simplă

```
if   (expresie)
          instructiune_1;
 else
          instructiune_2;
```

Structura alternativă generalizată

```
switch (selector)
{
case 1: instructiune_1;break;
case 2: instructiune_2;break;
---------------------------
case n: instructiune_n;break;
default : instructiune_i;
}
```

1. Se citeşte de la tastatură un număr natural întreg care reprezintă un an calendaristic. Să se verifice dacă numărul citit este un an bisect.

2. Un elev primeşte într-o zi trei note, nu toate bune. Se hotărăşte ca, dacă ultima notă este cel puţin 8, să le spună părinţilor toate notele primite iar dacă este mai mică decât 8, să le comunice doar nota cea mai mare dintre primele două. Introduceţi notele luate şi afişaţi notele pe care le va comunica părinţilor.

3. Se citesc trei numere întregi **x, y, z**. Dacă toate sunt pozitive să se afişeze cel mai mare număr dintre al doilea şi al treilea număr, în caz contrar să se afişeze suma primelor două numere.

4. Scrieţi un program care să testeze un caracter introdus de la tastatură. Dacă este literă mare, să afişeze mesajul **"Literă mare"**, dacă este literă mică să se afişeze mesajul **"Literă mică"**, altfel să se afişeze mesajul **"Nu este literă"**.

5. Se citeşte de la tastatură un număr natural întreg format din 3 cifre şi care afişează cel mai mic număr care se poate forma din cifrele sale. Modificaţi programul, astfel încât, să afişeze cel mai mare număr ce se poate forma din cifrele numărului citit.

6. Se introduc de la tastatură două numere întregi **a**, **b** şi un caracter **c** care reprezintă o operaţie aritmetică. Să se afişeze operaţia efectuată de operatorul citit şi să se calculeze valoarea lui **e** definită ca rezultat al aplicării operatorului aritmetic pe numerele **a** şi **b** (2 metode: se folosesc instrucţiuni **if-else** imbricate şi apoi cu instrucţiunea **switch-case**).

7. Scrieţi un program care să permită alegerea unei opţiuni dintr-un meniu afişat pe ecran, apoi se alege o operaţie din meniu prin introducerea numărului de ordine. Meniul conţine:

1. ORDONARE CRESCĂTOARE
2. ORDONARE DESCRESCĂTOARE

Programul ordonează crescător şi descrescător cifrele unui număr întreg format din 3 cifre.

8. Se citesc trei numere întregi nenule **a, b şi c** care reprezintă coeficienţii unei ecuaţii de gradul II. Să se rezolve ecuaţia. Testaţi programul pentru următoarele seturi de intrare:(1,-5,6), (1,-2,1), (1,1,1).

STRUCTURA REPETITIVĂ WHILE

Sintaxa acestei instrucţiuni este:

```
while (expresie)
{
  instrucţiuni;
}
```

Această instrucţiune se execută astfel:

- PAS 1: se evaluează expresia;
- PAS 2: dacă rezultatul este diferit de 0, adică corespunde valorii logice adevărat, atunci se execută instrucţiunile şi se revine la primul pas; altfel se trece la execuţia instrucţiunii care urmează instrucţiunii while.

EXEMPLE

a)Se citesc de la tastatură mai multe numere până la întâlnirea valorii 0. Să se scrie un program C++ care calculează şi afişează pe ecran suma numerelor pare şi produsul numerelor impare.

```
#include<iostream>
using namespace std;
int main()
{
    int n,s,p;
    cout<<"n=";cin>>n;
    s=0;p=1;
    while(n>0)
    {
        if(n % 2 == 0)
            s=s+n;
        else
            p=p*n;
    cout<<"n=";cin>>n;
    }
    cout<<"Suma numerelor pare este "<<s<<endl;
    cout<<"Produsul numerelor impare este "<<p;
}
```

b) Se citesc de la tastatură mai multe numere până la întâlnirea valorii 0. Să se scrie un program C++ care determină valoarea maximă şi de câte ori apare în şir.

```
#include<iostream>
using namespace std;
int main()
{
    int n,max,contor;
    cout<<"n=";cin>>n;
    max=n;contor=1;
    while(n>0)
    {
        cout<<"n=";cin>>n;
        if(max==n)
            contor++;
        else
            if(n>max)
            {
                max=n;
                contor=1;
            }
    }
cout<<"Valoarea maxima din sir este "<<max;
cout<<" si apare de "<<contor<<" ori.";
}
```

Explicaţii:

- se citeşte primul număr , se atribuie variabilei max valoarea primului număr citit şi se iniţializează contorul cu 1;
- se citeşte următorul număr;
- dacă max este egal cu n atunci se incrementează contorul;
- dacă n este mai mare decât max, atunci variabilei max i se atribuie valoarea lui n şi se iniţializează contorul cu 1.
- aceste operaţii se execută până când citim valoarea 0.

STRUCTURA REPETITIVĂ FOR

Sintaxa acestei instrucţiuni este:

```
for (exp1;exp2;exp3)
{
instrucţiuni;
}
```

- exp1, **pentru iniţializare**, prin care se stabileşte starea dinainte de prima execuţie a instrucţiunii;
- exp2, **pentru testare**, compară starea curentă cu starea care termină procesul de terminare; are rolul de a termina executarea repetată a instrucţiunilor;
- exp3, **pentru modificare**, prin schimbarea stării curente, astfel încât să se avanseze către starea finală.

Instrucţiunea **for** se execută astfel:

- PAS 1: se evaluează expresia **exp1**;
- PAS 2: se evaluează expresia **exp2**; dacă rezultatul este diferit de 0, adică

corespunde valorii logice adevărat, atunci se execută instrucţiunile; altfel se trece la execuţia instrucţiunii care urmează instrucţiunii for.

- PAS 3: se evaluează expresia **exp3** şi se revine la PAS 2.

EXEMPLE

a)Se citesc de la tastatură **n** numere întregi. Să se calculeze şi să se afişeze pe ecran suma numerelor pare şi produsul numerelor impare.

b)Înlocuiţi structura repetitivă FOR cu structura repetitivă WHILE.

```
#include<iostream>
using namespace std;
int main()
{
    int n,s,p,i,a;
    cout<<"n=";cin>>n;
    s=0;p=1;
    for(i=1;i<=n;i++)
    {
        cout<<"a=";cin>>a;
        if(a % 2==0)//sau !(a % 2)
            s=s+a;//sau s+=a;
        else
            p=p*a;//sau p*=a;
    }
cout<<"Suma numerelor pare este "<<s<<endl;
cout<<"Produsul numerelor impare este "<<p;
}
```

c)Se citesc de la tastatură **n** numere întregi. Să se calculeze media aritmetică a numerelor impare.

d)Înlocuiţi structura repetitivă FOR cu structura repetitivă WHILE.

```
#include<iostream>
using namespace std;
int main()
{
    int n,s,i,a,contor;
    cout<<"n=";cin>>n;
    s=0;contor=0;
    for(i=1;i<=n;i++)
    {
        cout<<"a=";cin>>a;
        if(a % 2!=0)
            {
                s=s+a;
                contor++;
            }
    }
cout<<"Media aritmetica a numerelor ";
cout<<"impare este "<<(float)s/contor<<" .";
}
```

STRUCTURA REPETITIVĂ DO.....WHILE

Sintaxa acestei instrucţiuni este:

```
do
{
  instrucţiuni;
}
while(expresie);
```

Această instrucţiune se execută astfel:
PAS 1: se execută instrucţiune;
PAS 2: se evaluează expresia;dacă rezultatul este diferit de 0, adică corespunde valorii logice adevărat, atunci se revine la primul pas; altfel se trece la execuţia instrucţiunii care urmează instrucţiunii do……while.

Spre deosebire de instrucţiunea WHILE instrucţiunea DO...WHILE se execută cel puţin o dată.

EXEMPLE

a)Se citesc cifrele unui număr începând cu cifra cea mai semnificativă. Să se afişeze numărul obţinut.

b)Modificaţi programul, înlocuind instrucţiunea DO…WHILE cu instrucţiunea WHILE.

```
#include<iostream>
using namespace std;
int main()
{
    long n;
    int cifra;
    cout<<"Cifra:";cin>>cifra;
    n=0;
    if(cifra>=0 && cifra<=9)
    do
    {
        n=n*10+cifra;
        cout<<"Cifra:";
        cin>>cifra;
    }
    while(cifra>=0 && cifra<=9);
cout<<"Numarul obtinut este "<<n<<" .";
}
```

PROBLEMĂ REZOLVATĂ

1.Se citeşte câte un caracter, până la întâlnirea caracterului @. Să se afişeze câte litere mari au fost introduse, câte litere mici, câte cifre şi câte alte caractere.

2.Modificaţi programul astfel încât, pentru fiecare caracter citit, să se afişeze un mesaj care să indice dacă s-a citit o literă mare, o literă mică, o cifră sau un alt caracter.

```
#include<iostream>
using namespace std;
int main()
{
    int lit_mica=0,lit_mare=0,cifra=0,alt_caracter=0;
    char c;
    cout<<"Introduceti caracterul:";cin>>c;
    while(c!='@')
    {
        if(c>='a' && c<='z')
            lit_mica++;
        else
            if(c>='A' && c<='Z')
                lit_mare++;
            else
                if(c>='0' && c<='9')
                    cifra++;
                else
                    alt_caracter++;
    cout<<"Introduceti caracterul:";cin>>c;
    }
    cout<<"Ai introdus "<<endl;
    cout<<lit_mica<<" litere mici"<<endl;
    cout<<lit_mare<<" litere mari"<<endl;
    cout<<cifra<<" cifre si "<<endl;
    cout<<alt_caracter<<" alte caractere."<<endl;
}
```

ALGORITMI FUNDAMENTALI

Aceşti algoritmi au fost concepuţi spre a veni în ajutorul programatorilor, care îi folosesc ori de câte ori este necesar în probleme, fără a mai fi nevoie să-i elaboreze de fiecare dată. Aceştia se referă la separarea cifrelor unui număr (folosit de fiecare dată când în rezolvarea unei probleme este necesar accesul la cifrele unui număr), determinarea divizorilor proprii ai unui număr natural dat, testarea dacă un număr natural mai mare ca 1 este prim, determinarea celui mai mare divizor comun a două numere naturale date, descompunerea unui număr natural în factori primi, determinarea maximului/minimului unui şir de numere citite, pe rând, de la dispozitivul de intrare.

1.Separarea cifrelor unui număr

Se va folosi rezultatul din matematică conform căruia restul împărţirii la 10 al unui număr întreg pozitiv îl reprezintă ultima cifră a numărului (cea mai puţin semnificativă), iar câtul împărţirii la 10, numărul fără ultima cifră. Repetând această operaţie cât timp numărul mai are cifre de separat, obţinem la fiecare pas o cifră a numărului, care poate fi prelucrată, de fiecare dată câtul obţinut devenind deîmpărţit. În algoritm, marcarea încheierii separării cifrelor se face când numărul dat devine 0, deci nu mai sunt cifre de separat.

Exemplu: n=2954

operaţia	cât	rest
2954:10	295	4
295:10	29	5
29:10	2	9
2:10	0	2

Limbajul C++

```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cout<<"n=";cin>>n;
    while(n)
    {
        cout<<n%10<<" ";
        n=n/10;
    }
}
```

Am obţinut cifrele numărului în ordine inversă: 4, 5, 9, 2.

2.Determinarea divizorilor proprii ai unui număr natural dat

De exemplu, dacă n=50, divizorii proprii sunt: 2, 5, 10, 25;
dacă n=45, divizorii proprii sunt: 3, 5, 9, 15;
dacă n=32, divizorii proprii sunt: 2, 4, 8, 16.
Putem continua cu exemplele, dar şi din acestea se poate observa că:
-cel mai mic divizor propriu posibil este 2
-cel mai mare divizor propriu posibil este jumătatea numărului [n/2]
Este suficient să testăm care din valorile cuprinse între 2 şi [n/2] împart exact numărul n dat şi astfel identificăm, pe rând, divizorii proprii ai numărului, care vor putea fi prelucraţi conform cerinţelor enunţului.
Exemple cu cele trei structuri repetitive:

```
#include<iostream>
using namespace std;
int main()
{
    int n,d;
    cout<<"n=";cin>>n;
    d=2;
    while(d<=n/2)
    {
        if(n%d==0)
            cout<<d<<" ";
        d++;
    }
}
```

```
#include<iostream>
using namespace std;
int main()
{
    int n,d;
    cout<<"n=";cin>>n;
    d=2;
    do
    {
        if(n%d==0)
            cout<<d<<" ";
        d++;
    }
    while(d<=n/2);
}
```

```
#include<iostream>
using namespace std;
int main()
{
    int n,d;
    cout<<"n=";cin>>n;
    for(d=2;d<=n/2;d++)
        if(n%d==0)
            cout<<d<<" ";
}
```

3.Testul de număr prim

Matematica ne spune că un număr este prim dacă are doar doi divizori, pe 1 şi numărul însuşi, deci când nu are divizori proprii. Sunt mai multe modalităţi de a verifica dacă un număr dat este prim sau nu. Noi o vom folosi pe cea conform căreia dacă numărul nu are divizori proprii atunci este prim, în caz contrar, dacă are cel puţin un divizor propriu, atunci numărul nu este prim.

```
#include<iostream>
using namespace std;
int main()
{
    int n,d,ok=1;
    cout<<"n=";cin>>n;
    for(d=2;d<=n/2 && ok==1;d++)
        if(n%d==0)
            ok=0;
        if(ok!=0)
            cout<<"Numarul este prim.";
        else
            cout<<"Numarul nu este prim.";
}
```

4.Determinarea celui mai mare divizor comun a două numere naturale

Algoritmul lui Euclid

Să presupunem că avem două numere naturale a şi b, pentru care trebuie să aflăm cel mai mare divizor comun(cmmdc).Se reţine în variabila r restul împărţirii lui a la b.Variabila a ia valoarea variabilei b iar b ia valoarea restului obţinut în urma împărţirii lui a la b. Aceste operaţii se execută cât timp b este diferit de 0. Cel mai mare divizor comun va fi variabila a.

```
#include<iostream>
using namespace std;
int main()
{
    int a,b,r;
    cout<<"a=";cin>>a;
    cout<<"b=";cin>>b;
    while(b)
    {
        r=a%b;
        a=b;
        b=r;
    }
    cout<<"cmmdc este "<<a;
}
```

Algoritmul scăderilor repetate

Algoritmul este următorul: cât timp cele două numere a şi b sunt diferite între ele, se scade din numărul mai mare numărul mai mic. În momentul în care cele două numere devin egale, cmmdc se află în oricare din cele două numere a sau b.

```
#include<iostream>
using namespace std;
int main()
{
    int a,b;
    cout<<"a=";cin>>a;
    cout<<"b=";cin>>b;
    while(a!=b)
    {
        if(a>b)
            a=a-b;
        else
            b=b-a;
    }
    cout<<"cmmdc este "<<a;
}
```

5.Descompunerea în factori primi a unui număr natural

Exemple

120	2
60	2
30	2
15	3
5	5
1	

45	3
15	3
5	5
1	

Algoritm:
se porneşte de la primul factor prim posibil, 2;
cât timp numărul dat este diferit de 1, se execută operaţiile:
dacă factorul îl divide pe n iniţializăm contorul
cât timp numărul se împarte exact la un factor prim
incrementăm contorul cu 1
se execută împărţirea, câtul devine deîmpărţit
se trece apoi la următorul factor prim

```
#include <iostream>
using namespace std;
int main()
{
    int n,d=2,k;
    cout<<"n=";cin>>n;
    while(n!=1)
    {
        if(n%d==0)
        {
            k=0;
            while(n%d==0)
            {
                k++;
                n=n/d;
            }
            cout<<d<<"^"<<k<<endl;
        }
        d++;
    }
}
```

6.Determinarea valorii minime/maxime dintr-un şir de numere

Determinarea valorii maxime

Se presupune că primul număr citit este maximul. Se citesc apoi, pe rând, numerele şi la fiecare pas se compară numărul citit cu maximul existent. Dacă numărul citit este mai mare decât maximul, se înlocuieşte maximul.

Determinarea valorii minime

Se presupune că primul număr citit este minimul. Se citesc apoi, pe rând, numerele şi la fiecare pas se compară numărul citit cu minimul existent. Dacă numărul citit este mai mic decât minimul, se înlocuieşte minimul.

```
#include<iostream>
using namespace std;
int main()
{
    int n,a,min,i;
    cout<<"n=";cin>>n;
    cout<<"a=";cin>>a;
    min=a;
    for(i=2;i<=n;i++)
    {
        cout<<"a=";cin>>a;
        if(a<min)
            min=a;
    }
    cout<<min;
}
```

```
#include<iostream>
using namespace std;
int main()
{
    int n,a,max,i;
    cout<<"n=";cin>>n;
    cout<<"a=";cin>>a;
    max=a;
    for(i=2;i<=n;i++)
    {
        cout<<"a=";cin>>a;
        if(a>max)
            max=a;
    }
    cout<<max;
}
```

Cuprins