

Cuprins

NOȚIUNI TEORETICE DESPRE LIMBAJUL C++ - PREZENTARE GENERALĂ	4
ELEMENTELE DE BAZĂ ALE LIMBAJULUI C++	5
Vocabularul limbajului	5
a)Setul de caractere	5
b)Identificatori. Cuvinte cheie	5
c)Comentarii.....	5
Scrierea și citirea în limbajul C++	5
Tipuri de date, constante, variabile	6
a)Tipuri de date.....	6
b)Constante	7
c)Variabile.....	7
Expresii. Operatori C++	8
a)Operatori aritmetici	8
b)Operatori relaționali	9
c)Operatori de incrementare și decrementare	9
d)Operatorii logici	10
e)Operatorii de asignare(atribuire)	10
f)Operatorul condițional.....	11
g)Operatorul de conversie explicită.....	11
h)Operatorul dimensiune sizeof.....	11
FIȘĂ DE LUCRU - OPERATORI C++	12
FIȘĂ DE LUCRU – STRUCTURA ALTERNATIVĂ.....	14
STRUCTURA REPETITIVĂ WHILE.....	15
STRUCTURA REPETITIVĂ FOR.....	16
STRUCTURA REPETITIVĂ DO..... WHILE	18
PROBLEMĂ REZOLVATĂ	18

ALGORITMI FUNDAMENTALI	19
1.Separarea cifrelor unui număr.....	19
2.Determinarea divizorilor proprii ai unui număr natural dat.....	20
3.Testul de număr prim	21
4.C.M.M.D.C.....	21
Algoritmul lui Euclid	21
Algoritmul scăderilor repetate	21
5.Descompunerea în factori primi a unui număr natural	22
6.Determinarea valorii minime/maxime dintr-un șir de numere	22
Determinarea valorii maxime	22
Determinarea valorii minime.....	22

NOȚIUNI TEORETICE DESPRE LIMBAJUL C++ - PREZENTARE GENERALĂ

Prin **programare** se înțelege în mod generic *transpunerea unor operații 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 și anume cea de trecere de la problema reală la transpunerea într-un limbaj de programare.
- o a doua etapă, automată, care transpune **codul sursă** (înșiruirea de instrucțiuni 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 operațiuni;
- și să livreze **datele de ieșire** (adică rezultatele).

Un **program** scris în limbajul C++ este compus din unul sau mai multe **fișiere sursă**. Un fișier sursă este un fișier text care conține codul sursă al unui program. Fiecare fișier sursă conține una sau mai multe **funcții** și eventual, referințe către unul sau mai **fișiere header**. Funcția principală a unui program este numită **main**.

Exemplu

```
//fișierul header iostream pentru operațiile de scriere și citire
#include <iostream>
/*fișierul header cmath folosit pentru a apela
  funcții matematice */
#include <cmath>
using namespace std;
//funcția principală main care poate apela alte funcții
int main()
{
    //partea declarativa-variabila a de tip întreg
    int a;
    //și variabila b de tip real
    float b;
    //scrierea lui a - apare pe ecran a=
    cout<<"a=";
    //citirea lui a de la tastatură
    cin>>a;
    //instrucțiune
    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 semnificație bine stabilită, numite *cuvinte cheie*.

Observație: când alegeți denumiri de variabile pentru programe **să nu utilizați** 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 explicații scurte sau observații 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):

- ✎ mulțimea de valori pe care variabila respectivă le poate lua
- ✎ cât și setul de operații pe care programatorul le poate efectua cu acea variabilă.

TIP	CARACTERISTICI
CHAR	reține un singur caracter Exemple: 'A','a','% ', etc.
INT	reține numere întregi cu semn Exemple: 23,-45,0, etc.
FLOAT	reține numere reale în format cu virgulă mobilă, în simplă precizie Exemple: 7.8965, -4.123, 7.0, etc.
DOUBLE	reține 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 inițial, dar care poate fi specificat ulterior declarării.

Modificatorii de tip. Limbajul C++ oferă pe lângă 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 fața tipului respectiv.*

b) Constante

Sunt date a căror valoare nu poate fi modificată în timpul execuției programului. Ele reprezintă un tip și o valoare și astfel pot fi de mai multe tipuri:

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

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

constantă caracter este de fapt un caracter între apostrofuri. Se reprezintă pe 8 biți, 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')

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

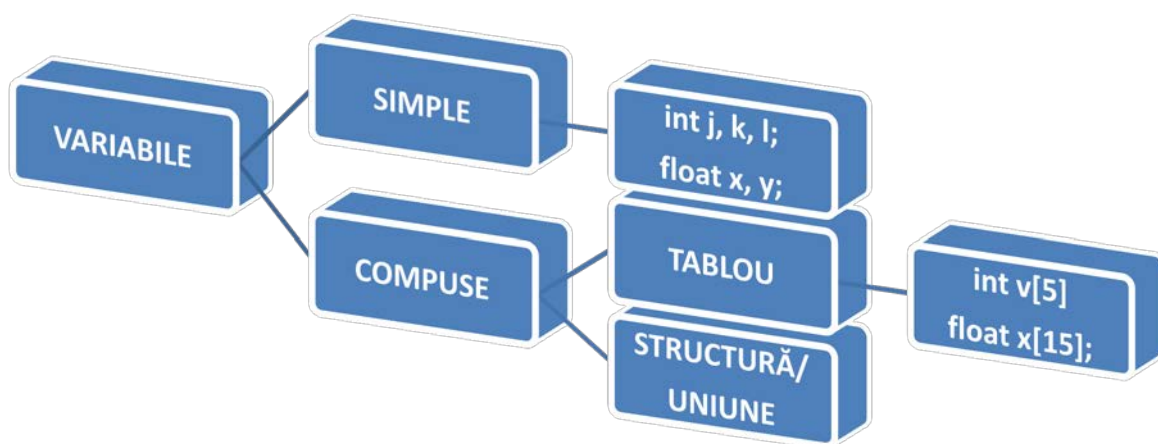
c) Variabile

Pentru a putea utiliza informațiile ce pot fi prelucrate prin intermediul programelor, trebuie să folosim denumiri (identificatori), care să fie compuși din - litere, cifre și liniuța 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 și tipul de date asociat. Exemple:

<i>int a</i>	variabila a este de tip întreg
<i>float b</i>	variabila b este de tip real
<i>char d</i>	variabila d este de tip caracter
<i>void</i>	variabila fără tip

Limbaajul C++ este *case sensitive*, adică face diferența dintre literele mici și mari, astfel încât, două denumiri de variabile sau de funcții, 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 funcții diferite. Variabilele pot fi:



Expresii. Operatori C++

Expresia este alcătuită dintr-unul sau mai mulți **operanzi** legați între ei prin **operatori** pentru a efectua anumite operații (calculare, atribuire, 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) Operatorii aritmetici

Nr. crt.	Operator	Semnificație	Tipul datelor	Exemplu
1.	+	adunare	numeric	$z = y + x;$
2.	-	scădere	numeric	$z = y - x;$
3.	*	înmulțire	numeric	$z = y * x;$
4.	/	împărțire	numeric	$z = y / x;$
5.	%	modulo	întreg	$z = y \% x;$

Operatorii +, -, /, * pot fi utilizați cu orice tip de dată, pe când operatorul **modulo** poate fi folosit doar cu datele de tip întreg. Operatorul **modulo** reține restul unei împărțiri 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	Operație
1.	<	mai mic
2.	>	mai mare
3.	<=	mai mic sau egal
4.	>=	mai mare sau egal
5.	==	egal
6.	!=	diferit

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

c) Operatori de incrementare și decrementare

Operatorul de incrementare (++) și **decrementare** (--) mărește sau micșorează valoarea operandului său cu 1 (unu). Operatorii de incrementare și decrementare se pot aplica variabilelor, dar nu și constantelor, aceștia putând fi prefixați sau postfixați.

Prefixați	Postfixați
++ operand	operand --
-- operand	operand ++

Diferența dintre operatorii **postfixați** și operatorii **prefixați** este aceea că dacă operandul este **postfixat** atunci mai întâi se folosește valoarea lui nemodificată, iar apoi se aplică operatorul respectiv. Dacă operandul este **prefixat** atunci mai întâi se aplică operatorul respectiv și abia apoi se folosește 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	Operație
1.	!	negarea logică
2.	&&	ȘI 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 condițional

Operatorii condiționali 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 și tipul expresiei va fi *exp2*, altfel valoarea și tipul va fi *exp3*.

? : sunt **operatorii condiționali** și trebuie să fie folosiți 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 și 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 octeți 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 ";
}
```

FIȘĂ DE LUCRU - OPERATORI C++

1. Se citește un număr natural care reprezintă timpul exprimat în minute. Scrieți programul care afișează timpul exprimat în ore și secunde.
2. Scrieți un program care să testeze un caracter introdus de la tastatură. Dacă este literă mare, să afișeze mesajul "Literă mare", dacă este literă mică să se afișeze mesajul "Literă mică", altfel să se afișeze mesajul "Nu este literă".
3. Scrieți un program care citește de la tastatură un număr natural cu trei cifre și care afișează apoi numărul obținut prin eliminarea cifrei din mijloc.
4. Scrieți un program care citește de la tastatură un număr natural cu patru cifre și care afișează pe câte un rând cifrele numărului.
5. Scrieți un program care citește de la tastatură un număr natural cu patru cifre și care afișează numărul obținut prin eliminarea cifrei sutelor. Modificați programul, astfel încât, să afișeze numărul obținut prin eliminarea cifrei zecilor.
6. Se citesc de la tastatură trei numere a, b și c. Să se afișeze valoarea maximă. Modificați programul, astfel încât, să afișeze valoarea minimă.
7. Se citește 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 și c care reprezintă laturile unui triunghi oarecare. Să se calculeze aria triunghiului.
9. Indicați rezultatele pe care le afișează programul următor. Explicați obținerea 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. Scrieți un program care convertește gradele Celsius în grade Fahrenheit conform formulei $f = (9/5) * c + 32$. Datele se citesc de la tastatură.

11. Se introduc 2 numere, a și b și un număr k. Să se afișeze un mesaj dacă fracția a/b poate fi simplificată prin k.

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

13. Fie variabilele x,y și u de tipul int. Scrieți o instrucțiune care mărește valoarea variabilei u cu câtul întreg al împărțirii lui x la y ?

14. Scrieți 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 afișează în urma execuției secvenței de instrucțiuni de mai jos, dacă pentru n se citește 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;
}
```

FIȘĂ 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 citește 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 primește într-o zi trei note, nu toate bune. Se hotărăște ca, dacă ultima notă este cel puțin 8, să le spună părinților toate notele primite iar dacă este mai mică decât 8, să le comunice doar nota cea mai mare dintre primele două. Introduceți notele luate și afișați notele pe care le va comunica părinților.
3. Se citesc trei numere întregi x , y , z . Dacă toate sunt pozitive să se afișeze cel mai mare număr dintre al doilea și al treilea număr, în caz contrar să se afișeze suma primelor două numere.
4. Scrieți un program care să testeze un caracter introdus de la tastatură. Dacă este literă mare, să afișeze mesajul "Literă mare", dacă este literă mică să se afișeze mesajul "Literă mică", altfel să se afișeze mesajul "Nu este literă".
5. Se citește de la tastatură un număr natural întreg format din 3 cifre și care afișează cel mai mic număr care se poate forma din cifrele sale. Modificați programul, astfel încât, să afișeze cel mai mare număr ce se poate forma din cifrele numărului citit.
6. Se introduc de la tastatură două numere întregi a , b și un caracter c care reprezintă o operație aritmetică. Să se afișeze operația efectuată de operatorul citit și să se calculeze valoarea lui e definită ca rezultat al aplicării operatorului aritmetic pe numerele a și b (2 metode: se folosesc instrucțiuni **if-else** imbricate și apoi cu instrucțiunea **switch-case**).
7. Scrieți un program care să permită alegerea unei opțiuni dintr-un meniu afișat pe ecran, apoi se alege o operație din meniu prin introducerea numărului de ordine. Meniul conține:
 1. ORDONARE CRESCĂTOARE
 2. ORDONARE DESCRESĂTOARE
 Programul ordonează crescător și descrescător cifrele unui număr întreg format din 3 cifre.
8. Se citesc trei numere întregi nenule a , b și c care reprezintă coeficienții unei ecuații de gradul II. Să se rezolve ecuația. Testați programul pentru următoarele seturi de intrare: (1,-5,6), (1,-2,1), (1,1,1).

STRUCTURA REPETITIVĂ WHILE

Sintaxa acestei instrucțiuni este:

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

Această instrucțiune se execută astfel:

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

EXEMPLE

a) Se citesc de la tastatură mai multe numere până la întâlnirea valorii 0. Să se scrie un program C++ care calculează și afișează pe ecran suma numerelor pare și 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 și de câte ori apare în șir.

Explicații:

- se citește primul număr, se atribuie variabilei max valoarea primului număr citit și se inițializează contorul cu 1;
- se citește 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 și se inițializează contorul cu 1.
- aceste operații se execută până când citim valoarea 0.

```
#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.";
}
```

STRUCTURA REPETITIVĂ FOR

Sintaxa acestei instrucțiuni este:

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

- exp1, **pentru inițializare**, prin care se stabilește starea dinainte de prima execuție a instrucțiunii;
- exp2, **pentru testare**, compară starea curentă cu starea care termină procesul de terminare; are rolul de a termina executarea repetată a instrucțiunilor;
- exp3, **pentru modificare**, prin schimbarea stării curente, astfel încât să se avanseze către starea finală.

Instrucțiunea **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 instrucțiunile; altfel se trece la execuția instrucțiunii care urmează instrucțiunii for.

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

EXEMPLE

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

b) Înlocuiți 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) Înlocuiți 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 instrucțiuni este:

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

se

Această instrucțiune se execută astfel:

PAS 1: se execută instrucțiune;

PAS 2: se evaluează expresia; dacă rezultatul este diferit de 0, adică corespunde valorii logice adevărat, atunci se revine la primul pas; altfel

trece la execuția instrucțiunii care urmează instrucțiunii do.....while.

Spre deosebire de instrucțiunea WHILE instrucțiunea DO...WHILE se execută cel puțin o dată.

EXAMPLE

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

b) Modificați programul, înlocuind instrucțiunea DO...WHILE cu instrucțiunea 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 citește câte un caracter, până la întâlnirea caracterului @. Să se afișeze câte litere mari au fost introduse, câte litere mici, câte cifre și câte alte caractere.

2. Modificați programul astfel încât, pentru fiecare caracter citit, să se afișeze 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

Acești algoritmi au fost concepuți 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. Aceștia 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 șir de numere citite, pe rând, de la dispozitivul de intrare.

1.Separarea cifrelor unui număr

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

Exemplu: n=2954

operația	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 obținut 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 și 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 și $[n/2]$ împart exact numărul n dat și astfel identificăm, pe rând, divizorii proprii ai numărului, care vor putea fi prelucrați conform cerințelor enunțului.

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 și numărul însuși, deci când nu are divizori proprii. Sunt mai multe modalități 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 puțin 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 și b , pentru care trebuie să aflăm cel mai mare divizor comun (cmmdc). Se reține în variabila r restul împărțirii lui a la b . Variabila a ia valoarea variabilei b iar b ia valoarea restului obținut în urma împărțirii lui a la b . Aceste operații 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 și 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	45	3
60	2	15	3
30	2	5	5
15	3	1	
5	5		
1			

Algoritm:

se pornește de la primul factor prim posibil, 2;

cât timp numărul dat este diferit de 1, se

execută operațiile:

dacă factorul îl divide pe n inițializăm
contorul

cât timp numărul se împarte exact la un
factor prim

incrementăm contorul cu 1

se execută împărțirea, câtul devine deîmpărțit

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 șir de numere

Determinarea valorii maxime

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

Determinarea valorii minime

Se presupune că primul număr citit este minimul. Se citesc apoi, pe rând, numerele și la fiecare pas se compară numărul citit cu minimul existent. Dacă numărul citit este mai mic decât minimul, se înlocuiește 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;
}
```