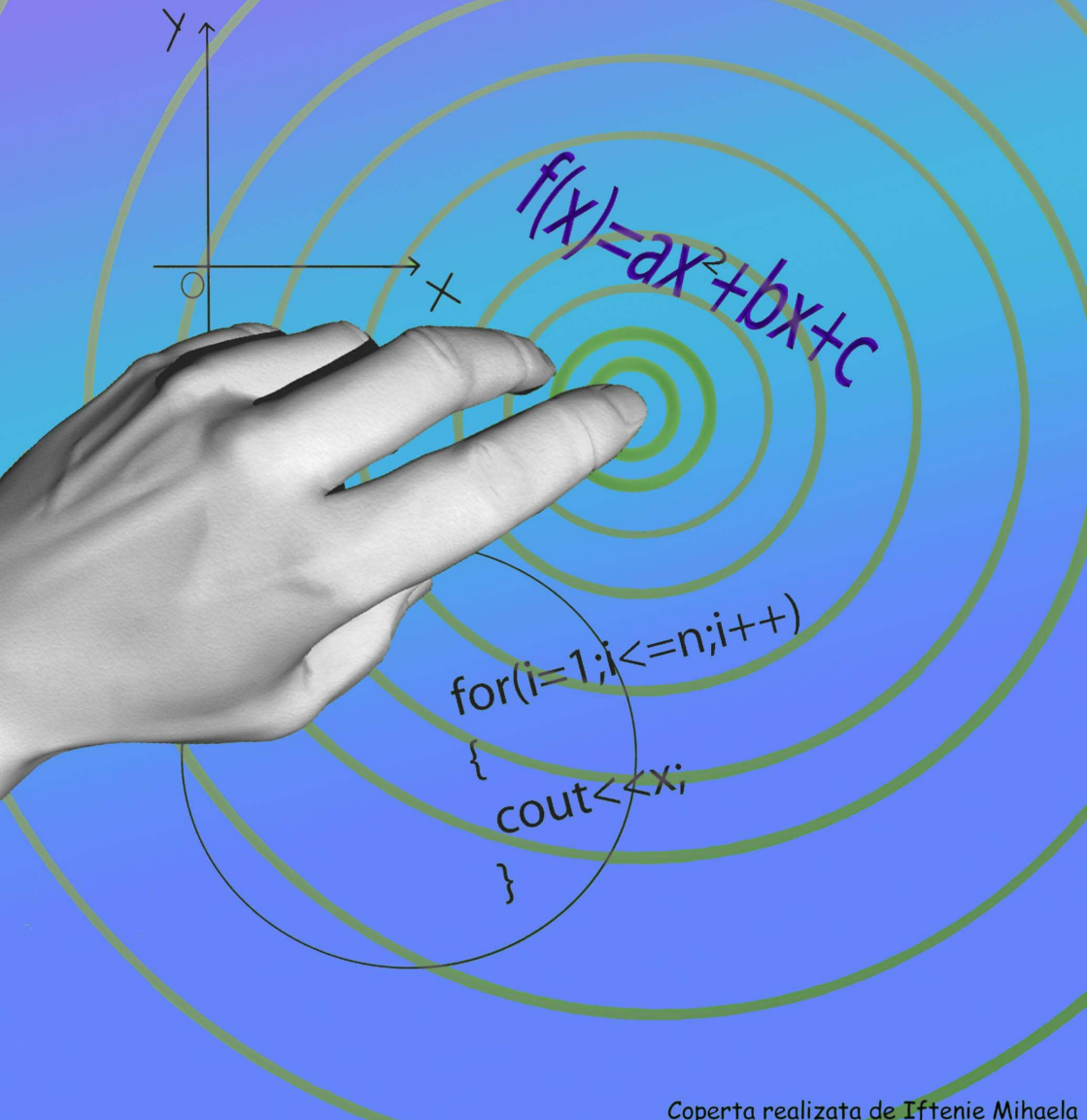
DIGITAL

Revista de Matematica-Informatica

Nr. 4 mai 2015



Coperta realizata de Iftenie Mihaela

DIGITAL
Revista de Matematica – Informatica

Nr.4/Mai 2015

ISSN 2068-3642

Informatica-Prezentare generală

Termenul **informatică** desemnează ştiinţa procesării sistematice a informaţiei, în special a procesării cu ajutorul calculatoarelor. Termenul englez corespunzător este Computer Science (stiinţa calculatoarelor).

Istoric, informatica s-a dezvoltat ca ştiinţă din matematică, în timp ce dezvoltarea primelor calulatoareîşi are originea în electrotehnică şi telecomunicaţii. De aceea, calculatorul reprezintă doar dispozitivul pe care sunt implementate conceptele teoretice.Informaticianul olandez Edsger Dijkstra afirma: "În informatică ai de-a face cu calculatorul, aşa cum ai în astronomie cu telescopul".Termenul **informatică** provine din alăturarea cuvintelor informaţie şi matematică. Alte surse susţin că provine din combinaţia informaţie şi automatică.În prezent, informatica îşi găseşte aplicaţii în toate domeniile vieţii.Prezenţa ei este puternic amplificată de impactul pe care îl are Internetul. Reţeaua la nivelmondial a revoluţionat comunicarea dintre companii, logistica, mass media, dar şi viaţa privată a fiecarui individ. Mai puţin vizibil, dar totuşi omniprezent, informatica şi-a câştigat un loc stabil până şi în aparatelecasnice, ca de exemplu video recorder-ul sau maşina de spălat, în care sunt înglobate aşa-numitele embedded Systems (sisteme inglobate), care asigură acestor aparate un comportament mai mult sau mai puţin "inteligent".

Computerele pot administra, proteja, transmite şi prelucra o mare cantitate de date într-un timp scurt.Pentru efectuarea unor astfel de operaţii este necesară o interacţiune complex între sistemele de hardware şi de software, care reprezintă domeniile fundamentale de cercetare în Informatică.

Ca sistem ştiinţific fundamental, informatica are, la fel ca şi matematica, implicaţii profunde în multe alte domenii ale ştiinţei. Dacă prin matematică se înţelegeun "sistem de gândire formal", atunci informatica se concentrează pe ceea ce este "formal realizabil", adicăceea ce este realizabil din punctul de vedere al maşinii.

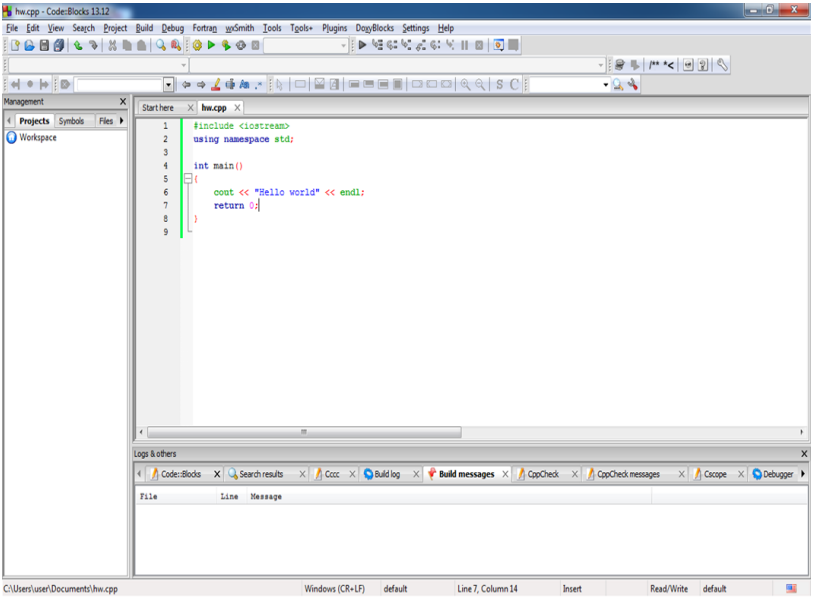
Informatician Bargan Gabriela
Liceul Teoretic "Callatis" Mangalia

Utilizare codeblocks

Code::Blocks este un IDE. El permite scrierea sursei, compilarea acesteia, lansarea în execuție a programului executabil. În mod tradițional, fiecare programator începător scrie programul hello world. Acest program nu face altceva decât să afișeze pe ecran mesajul hello world, dar este un bun exemplu de program, scris în C++.

Creare proiect

1. Create a newproject->Console Application->Go
2. C++ ->Next
3. Project Title: se da un nume proiectului, se stabileste locația proiectului->Finish
4. Sources->main.cpp// textul sursă se redactează direct în main.cpp
5. Redactare cod sursă, redenumire fișier cod sursă (main.cpp)



6. Urmează compilarea și, dacă ceasta reușește, urmează crearea executabilului și lansarea lui. Code::Blocks oferă mai multe opțiuni pentru aceste operații; dintre ele vă recomandăm folosirea opțiunii **Build -> Build and run**, cu scurtătura , care include toate aceste operații.

Depanare program (executarea unui program linie cu linie)

1. **Debug**->Edit watches->se adaugă, (ADD) variabilele ce vor fi urmărite în execuție->OK
2. Click pe prima linie de program->Debug->ToggleBreakpoint (F5) ->linia va fi marcată printr-o bulină roșie
3. Debug->Start(F8); dacă citirea se face de la tastatură, în fereastra Output care se deschide (fereastra consola) se introduc datele de intrare->Enter->Fereastra Output se minimizează (se pune pe bară, NU SE ÎNCHIDE); dacă citirea se face din fișier, fereastra Output se minimizează (de asemenea se pune pe bară, NU SE ÎNCHIDE)
4. Se vizualizează fereastra Watches: Debug ->DebuggingWindows->Watches (se bifează)
5. Se poziționează mouse în fereastra de editare (în main.cpp) pe linia marcată cu săgeată galbenă, care indică linia curentă (click cu mouse acolo, la început de linie)
6. Cu tasta F7 se trece la linia următoare, pas cu pas, și se urmărește în același timp, cum se modifică valorile variabilelor în fereastra Watches
7. Dacă programul conține funcții create de către utilizator și se dorește urmărirea instrucțiunilor în corpul funcțiilor, în loc de F7 se utilizează combinația Shift-F7

Prof. Informatică Gradul I
Magaz Georgeta-Liceul Teoretic „Callatis" Mangalia

Biblioteca graphics.h

Această bibliotecă este o rescriere a bibliotecii BGI din mediul Borland C++, pe care o îmbogăţeşte cu funcţii noi. Ea poate fi un punct de pornire pentru elevii care nu au mail ucrat cu funcţii grafice până acum. Pentru a utiliza această bibliotecă într-un proiect realizat în Code::Blocks trebuie să urmaţii paşii de mai jos:

1. Se crează un proiect nou de tip Empty Project (se urmează paşii Wizardului);
2. Se copiază în folderul proiectului biblioteca graphics.h
(Descărcaţi biblioteca de aici: http://www.uniqueness-template.com/devcpp/graphics.h)

3. Se adaugă opțiunile următoare în meniul Project ->BuildOptions... ->Linkersettings la OtherLinkerSettings:
-lbgi
-lgdi32
-lcomdlg32
-luuid
-loleaut32
-lole32

4. Se adaugă un fișier sursă nou în proiect, de exemplu:

```
#include <graphics.h>

int main()
{
initwindow(400,300); //open a 400x300 graphicswindow
moveto(0,0);
lineto(50,50);
lineto(50,100);
lineto(150,120);
while(!kbhit()); //wait for usertopress a key
closegraph(); //closegraphicswindow
return 0;
}
```

5. F9 şi Enjoy ! 

Prof. Informatică Gradul I
Georgeta Magaz-Liceul Teoretic „Callatis" Mangalia

Problemă propusă pentru clasa a XI(Aplicație pe liste simplu înlănțuite)

Intr-un copac sunt mere și pere. Fiecărei crengi îi corespunde un nod care conține drept informație numărul de mere si numărul de pere de pe creangă. O maimuță care se plimbă prin copac mănâncă de pe fiecare creanga fie un măr fie o pară. Maimuța se oprește pentru odihnă în momentul în care una din crengi se golește fie de mere fie de pere. Să se afișeze numărul de ordine al acelei crengi pe care s-a oprit să se odihnească maimuța. Ordinea în care se plimbă prin copac este cea inițială.

Creanga1 creanga 2 creanga 3 creanga 4 creanga 5

Mere pere mere pere mere pere mere pere mere pere

6 7 4 2 5 6 4 2 2 3

De pe creanga 1 mănâncă: mar ramin 5 mere si 7 pere

De pe creanga 2 mănâncă: para ramin 4 mere si o 1 para

De pe creanga 3 mănâncă: 1 mar si avem 4 mere si 6 pere

De pe creanga 4 mănâncă 1 para si avem 4 mere si 1 para

De pe creanga 5 mănâncă 1 mar si ramin 1 mar si 3 pere

De pe creanga 1 mănâncă para si ramin 5 mere s 6 pere

De pe creanga 2 mănâncă para si pentru bca avem 4 mere si o para ne oprim

Observație:

Se construiește o listă simplu înlănțuită circulară care se parcurge până în momentul în care se devine 0 fie numărul de mere fie numărul de pere.

Se continuă… .

```
#include<iostream.h>
struct Nod
        { intinfo;
          Nod* adr_urm;
        };
        Nod *v,*sf;

        voidAdaugare(Nod*& v,int val)
        {
        Nod* c;
        if (!v)
            {    v=new Nod; v->info=val;  v->adr_urm=0;
        sf=v;
            }
```

```
        }
    else
    {  c=new Nod; sf->adr_urm=c;
    c->info=val; c->adr_urm=0;
    sf=c;
     }
    }

    void Listare(Nod*  v)
    {
      Nod* c=v;

    while(c)
     {    cout<<c->info<<"  ";
      c=c->adr_urm;
      }
    cout<<endl;
    }

    void Inserare_dupa(Nod* v,  intval,int val1)
    {  Nod *c,*d;
      c=v;
    while   (c->info!=val)   c=c->adr_urm;
      d=new Nod; d->info=val1;   d->adr_urm=c->adr_urm; c-
>adr_urm=d;
    if   (d->adr_urm)     sf=d;
    }

    void Inserare_inainte(Nod*& v,int val,intval1)
    {  Nod *c,*d;
    If   (v->info==val)
          {
          d=new Nod;
    d->info=val1;   d->adr_urm=v;
          v=d;
          }
    else
       {  c=v;
    while(c->adr_urm->info!=val)  c=c->adr_urm;
         d= new Nod;    d->info=val1;    d->adr_urm=c->adr_urm;    c-
>adr_urm=d;
       }
    }
```

```
voidSterg(Nod*&v,int val)
{  Nod *c,*man;
if(v->info==val)
   {  man=v; v=v->adr_urm;   }
else
    {  c=v;
while(c->adr_urm->info!=val)  c=c->adr_urm;
man=c->adr_urm;c->adr_urm=man->adr_urm;
if   (man==sf)  sf=c;
     }
deleteman;
```

Prof. Informatică Gradul I
Pilat Elena-Mihaela-Liceul Teoretic „Callatis" Mangalia

Placa de bază

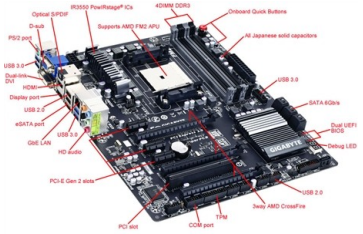
Placa de bază (engleză: *motherboard*) este **placa principală** a unui calculator, o componentă hardware complexă pe care sunt montate multe alte componente hardware ale PC-urilor sau ale altor aparate electronice computerizate.

Placa de bază conține și conectori electrici pentru comunicarea cu celelalte componente ale sistemului calculator. Pe placa de bază se conectează între ele unitatea centrală de prelucrare (UCP) și alte subsisteme și dispozitive electronice: interfețe, memorie, etc.

Un exemplu tipic de computer desktop are microprocesor, memorie RAM și alte componente esențiale conectate la placa de bază. Pot fi atașate și alte componente, cum ar fi dispozitive externe de stocare, controlere pentru placa video, placa de sunet și alte dispozitive periferice, deși calculatoarele moderne au integrate toate aceste periferice.

O componentă importantă a plăcii de bază este suportul pentru microprocesor chipset-ul, care are rolul de interfață de conectare între microprocesor, magistrala lui de date și diferitele componente externe. Acest chipset determină, într-o măsură, caracteristicile și capacitățile plăcii de bază. Plăcile de bază pot avea integrate pe ele și funcționalități cum ar fi cele de la:placă de sunet placă de rețea placă video
În mod normal aceasta este o soluție comodă și ieftină. Pentru o funcționalitate de bună calitate și viteză se folosesc însă plăci/cartele separate, care se conectează la placa de bază.

Placile de sunet,de retea,video care nu sunt încorporate în placa de bază se montează în porturile PCI (mai nou,placa video se montează în portul PCI-Express).Placa video se montează în portul PCI-Express,deoarece portul PCI-Express are o viteză de transmitere a datelor mai mare. Portul PCI-Express se află puțin mai în spate față de celelalte porturi PCI.



Informatician Bargan Gabriela-
Liceul Teoretic „Callatis” Mangalia

Problema propusa pentru clasa a IX-a

O matrice cu n linii și n coloane conține toate numerele de la 0 la $(n-1)^2$ în care suma elementelor de pe fiecare linie , coloana și diagonala este aceeași se numește pătrat magic.

Ex:

5 0 7
6 4 2
1 8 3

Să se scrie un program care să verifice printr-o singură parcurgere dacă o matrice cu n linii și n coloane este pătrat magic

Soluție:

```
#include<fstream.h>
int main(void)
{
	int v[50],mat,n,i,j;
	ifstream f("pb2.in");
	ofstream g("pb2.out");
	f>>n;
	for(i=0;i<50;i++)
		v[i]=0;
	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
		{
			f>>mat;
			v[j]+=mat;    //coloanele
			v[n+i]+=mat;  //liniile
			if(i==j)
				v[2*n]+=mat; //diagonala principală
			if(i==n-(j+1))
				v[2*n+1]+=mat; //diagonala secundară
		}
	j=0 ;     //observator
	for(i=0;i<n-1;i++)
		if(v[i]!=v[i+1])
			j=1;
	if(j)
		g<<"nu este pătrat magic";
	else
		g<<"este pătrat magic";
	f.close();
	g.close();
	return 0;
}
```

Prof. Informatică Gradul I
Pilat Elena Mihaela-LiceulTeoretic „Callatis” Mangalia

Problemă propusă pentru Clasa a IX-a

Fie A si B două mulțimi și relația dintre ele (O relație R este o submulțime a lui A X B cu alte cuvinte o relatie atașează unui element din A un element din B.)

EX:

A={1,2,3}

B={1,2} ➔**R={(1,1);(1,2);(3,1)}**

Să se verifice dacă există o funcție definită pe mulțimea A cu valori în mulțimea B.

```
#include<iostream.h>
intmain(void)
{
int i,j,m,n,p, dublura, A[20], B[20], R[3][50];
cout<<"m=";
cin>>m;
for (i=1;i<=m;i++)
                        cin>>A[i];
cout<<"n=";
cin>>n;
for (i=1;i<=n;i++)
                        cin>>B[i];
cout<<"p=";
cin>>p;
for (i=1;i<=2;i++)
                        for (j=1;j<=p;j++)
                                                cin>>R[i][j];
if (p!=m)
        cout<<"Relația nu este o funcție.";
else
        {
        dublura=1;
        for(i=1;(i<=m-1)&& (dublura);i++)
                                {
                        for(j=i+1;(j<=m)&& (dublura);j++)
                                                {
                                                if (R[1][i]==R[1]
[j])

                                                        dublura=0;

                                                }

        if (dublura==0)
                cout<<"Relația nu este funcție.";
        else
```

```
cout<<"Relația este funcție";
        }
}
```

Exemplul 1

m=3 A(1,2,3)

n=2 B(1,2)

p=3 212

313

Rezultat afișat „Nu este funcție”

Exemplul 2

m=3 A(1,2,3)

n=2 B(1,2)

p=3 123

121

Rezultat afisat „Este funcție”

Modelare Matematică

Relația R se poate reprezenta astfel:

$\begin{pmatrix} 1 & 1 & 3 \\ 1 & 2 & 1 \end{pmatrix}$ nu e funcție pentru că elementul 2 din mulțimea A nu are corespondent în mulțimea B.

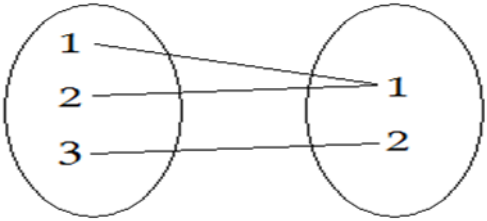
f : A→B este o relație cu proprietatea că (oricare) **a** aparține A daca există cel puțin un **b**

Relatia R se poate reprezenta astfel:

$\begin{pmatrix} 1 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix}$

nu e funcție pentru că elementul 2 din mulțimea A nu are corespondent în mulțimea B.

f : A→B este o relație cu proprietatea că (oricare) **a** aparține A dacă există cel puțin un b aparține B a.i. $f_{(a)}=b$



$$\begin{pmatrix} 1 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

este functie

Verificam daca numarul de elemente al multimii A este egal cu numarul de elemente al multimii B si printre elementele primei linii din reprezentarea lui R nu se afla dubluri.(acest lucru ne asigura ca ficare element din multimea A are cel putin un corespondent in multimea B

Prof. Informatica Gradul I
Pilat Elena Mihaela-Liceul Teoretic „Callatis" Mangalia

Cât bine face calculatorul copiilor?

Este benefic pentru fiecare copil să se familiarizeze cu calculatorul acesta fiind indispensabil pe viitor, insă parinţii sunt obligaţi să-l supravegheze, permiţându-i accesul în limita unui program bine stabilit si explicat.

Calculatorul dezvoltă capacitatea de concentrare şi atenţie, creşte capacitatea de reacţie la fel si coordonarea. Copilului timid îi oferă posibilitatea de a interacţiona cu cineva fără să-i fie teama de respingere. În acelaşi timp cu ajutorul calculatorului găsim o varietate de programe educative, prezentate mai atrăgător si pe înţelesul celui mic. Preşcolarul se va obişnui mai uşor cu literele şi vă învaţă să citească mult mai uşor, jocurile pe calculator îi pot dezvolta gâdirea şi logica matematică.

Cat poate dăuna calculatorul?

Studiile au arătat ca timpul petrecut în faţa calculatorului de către copii, nu ar trebui să depăşească una-două ore pe zi. Utilizarea îndelungată a PC-ului poate cauza dereglări ale sistemului nervos central - insomnii, cosmare noaptea, somn agitat, dependenţă. Jocurile pe calculator îl sustrag pe copil într-o lume imaginară, deschisă numai lui, înlocuiesc cu succes activităţile sociale specifice vârstei. În cazul unui copil retras, starea se poate agrava dispărand treptat nevoia de comunicare. Înainte de orice, sau înanite de achiziţionarea unui calculator pentru copii este de preferat să apelaţi mai intai la medicul dumneavostră pediatru în masură să vă prescrie ce este mai indicat pentru stadiul de dezvoltare al copilului dumneavoastră.

Daca îi permiteţi celui mic să se joace la calculator, trebuie să fiţi atenţi în alegerea jocurilor si programelor preferate. Nu-l lăsaţi pe copil să petreacă mult timp in faţa monitorului. Chiar dacă insistă explicaţi-i ferm că programul de joaca nu se discută. Încercaţi să înlocuiţi calculatorul cu alte activităţi: plimbări în aer liber, jocuri colective, etc.

Informatician Bargan Gabriela-
Liceul Teoretic „Callatis" Mangalia

Problemă propusă pentru clasa a IX-a

O matrice cu n linii și n coloane conține toate numerele de la 0 la $(n-1)^2$ în care suma elementelor de pe fiecare linie , coloană și diagonală este aceeași se numește pătrat magic.

Ex:

5 0 7
6 4 2
1 8 3

Să se scrie un program care să verifice printr-o singură parcurgere dacă o matrice cu n linii si n coloane este pătrat magic

Soluție:

```
#include<fstream.h>
int main(void)
{
        int v[50],mat,n,i,j;
        ifstream f("pb2.in");
        ofstream g("pb2.out");
        f>>n;
        for(i=0;i<50;i++)
                v[i]=0;
        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                {
                        f>>mat;
                        v[j]+=mat;    //coloanele
                        v[n+i]+=mat;  //liniile
                        if(i==j)
                                v[2*n]+=mat; //diagonala principală
                        if(i==n-(j+1))
                                v[2*n+1]+=mat; //diagonala secundară
                }
        j=0 ;    //observator
        for(i=0;i<n-1;i++)
                if(v[i]!=v[i+1])
                        j=1;
        if(j)
                g<<"nu este pătrat magic";
        else
                g<<"este pătrat magic";
        f.close();
        g.close();
        return 0;
}
```

Prof. Informatică Gradul I
Pilat Elena Mihaela-Liceul Teoretic „Callatis" Mangalia

Grafuri Neorientate

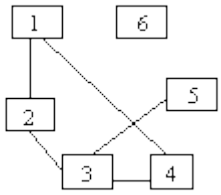
Graf neorientat= o pereche de multimi **=(V, E)**unde **V** este o mulţime finită nevida de elemente numite noduri iar **E** o multime de perechi neordonate din **V,** numite muchii. Notăm graful cu **G=(V, E)**.

Intr-un un graf **G=(V, E) neorientat** relaţia binară este simetrică: **(v,w)ÎE** atunci **(w,v) ÎE**.

Nod = element al mulţimii **V**, unde **G=(V, E)** este un graf neorientat.

Muchie = element al mulţimii **E** ce descrie o relaţie existentă între două noduri din **V**, unde **G=(V, E)** este un graf neorientat;

In figura alaturata:



V={1,2,3,4,5,6} sunt noduri
E={[1,2], [1,4], [2,3],[3,4],[3,5]} sunt muchii
G=**(V, E)**=({1,2,3,4,5,6}, {[1,2], [1,4], [2,3], [3,4],[3,5]})

Adiacenta: Într-un graf neorientat existenţa muchiei **(v,w)** presupune că **w** este adiacent cu **v** şi **v** adiacent cu **w**.

În exemplul din figura de mai sus vârful **1** este adiacent cu **4** dar **1** şi **3** nu reprezintă o pereche de vârfuri adiacente.

Incidenţă = o muchie este incidentă cu un nod dacă îl are pe acesta ca extremitate. Muchia **(v,w)** este incidentă în nodul **v** respectiv **w**.

Grad = Gradul unui nod **v**, dintr-un graf neorientat, este un număr natural ce reprezintă numărul de noduri adiacente cu acesta (sau numarul de muchii incidente cu nodul respectiv)

Nod izolat = Un nod cu gradul **0.**

Nod terminal= un nod cu gradul **1**

Daca un graf neorientat are m muchii atunci suma gradelor tuturor nodurilor este 2m

In orice graf G exista un numar par de noduri de grad impar

Elev Potamianos Cătălin-Gabriel Clasa a XI a B
Liceul Teoretic "Callatis" Mangalia

MATEMATICĂ

Grupuri

Un cuplu $(G, \circ)$, format dintr-o mulțime nevidă G și o lege de compoziție internă " $\circ$ " pe G, este ***grup*** dacă sunt satisfăcute axiomele:

Axioma închiderii

1.Oricare ar fi x și y din G, și rezultatul operației $x \circ y$ face parte din G

Axioma asociativității

1.Oricare ar fi x,y,z din G, $(x \circ y) \circ z = x \circ (y \circ z)$

Axioma elementului neutru

1.Există un element e în G, astfel încât $e \circ x = x \circ e = x$, oricare ar fi x din G

Axioma elementelor simetrice

1.Oricare ar fi x din G, există y în G cu proprietatea că $x \circ y = y \circ x = e$

2.Dacă este satisfăcută și axioma

Axioma comutativității

1.Oricare ar fi x,y din G, $x \circ y = y \circ x$

2.atunci grupul $(G, \circ)$ se numește *grup* ***comutativ*** sau ***abelian***.

Prof Matematică Gradul I

Faliboga Lucian Scoala Generala "Gala Galaction" Mangalia

Morfisme şi izomorfisme

I.Morfisme

În matematică, o funcție f : G → G' se numește **morfism de grupuri** în următoarele condiții: G și G' admit fiecare o structură de grup, cu operațiile notate • și respectiv $\circ$, iar f(x•y) = f(x) $\circ$ f(y), $\forall$ x,y $\in$ G.

Proprietăți:

1. Dacă e și e' sunt elementele neutre ale lui G si G' atunci f(e)=e'.
2. $\forall$ x $\in$ G, $f(x^{-1}) = (f(x))^{-1}$.
3. θ : G → G', θ(x)=e', x $\in$ G este evident morfism de grupuri numit morfismul nul.
4. Compunerea de morfisme de grupuri este tot un morfism de grupuri.
5. 1_G : G → G, 1_G(x) = x, x $\in$ G este evident morfism de grupuri numit morfismul identic al grupului G. În plus, dacă f : G → G' este morfism de grupuri atunci au loc: $f \circ 1_G = f$ și $1_{G'} \circ f = f$.

f este izomorfism de grupuri dacă și numai dacă f este bijecție.

II.Izomorfisme

În matematică, prin **izomorfism** (din limba greacă : ἴσος isos "egal", și μορφή morphe "formă") se înțelege o funcție între două mulțimi peste care s-au definit câte o structură algebrică, funcție care satisface două condiții:

1.este morfism (adică păstrează structura algebrică, în sensul că orice relație ar exista între niște elemente din prima mulțime, relația respectivă se regăsește între elementele corespunzătoare - imagini prin funcția studiată - din a doua mulțime),

2.admite un alt morfism care o "inversează" (formal, pentru f:A→B, să existe g:B→A *morfism* astfel încât $g \circ f = 1_A$ și $f \circ g = 1_B$). Notă: această condiție necesită ca f să fie bijectivă, dar cere în plus ca inversa ei să fie tot morfism.

Prof Matematică Gradul I
Faliboga Lucian Scoala Generala "Gala Galaction" Mangalia

Inele

Un **inel** $I = (A, +, *)$ este o structură algebrică formată dintr-o mulțime suport A și două operații binare, definite pe produsul cartezian $A \times A$ cu valori în A, numite convențional $+$ (sau operația aditivă) și $*$ (sau operația multiplicativă), astfel încât:

$G = (A, +)$ formează un grup comutativ sau abelian. Elementul neutru al lui G se notează în general cu 0.

$S = (A, *)$ formează un monoid.

Se îndeplinește proprietatea de distributivitate a înmulțirii față de adunare, adică pentru orice $x, y, z \in A$:

$$x * (y + z) = (x * y) + (x * z)$$
$$(x + y) * z = (x * z) + (y * z)$$

Termenul a fost introdus în 1897 de David Hilbert.

Dacă operația de înmulțire este comutativă, adică

$(\forall x, y \in A)\ x * y = y * x$ atunci inelul A este un ***inel comutativ***.

Dacă $A \neq \{0\}$ și înmulțirea admite element neutru, adică

$(\exists 1 \in A)\ (\forall x \in A)\ 1 * x = x * 1 = x$ atunci inelul A este ***inel cu unitate*** sau ***inel unitar***.

Un inel comutativ cu cel puțin două elemente și fără divizori ai lui zero se numește inel integru (sau domeniu de integritate).

Prof Matematică Gradul I
Ghiulserin Sali Liceul Teoretic "Callatis" Mangalia

Corpuri

Se numește **corp** un triplet $(K,+,*)$ în care K este o mulțime cu cel puțin două elemente , iar $+$ și $*$ două operații pe K (numite „adunare" respectiv „înmulțire") satisfăcând trei axiome:

$(K,+)$ este grup abelian cu elementul neutru notat cu 0.

$(K\setminus\{0\},*)$ este grup cu elementul neutru notat cu 1.

1. Înmulțirea este distributivă față de adunare, adică pentru orice $x,y,z\in K$:

$$x*(y+z)=x*y+x*z$$

$$(y+z)*x=y*x+z*x$$

Grupul $(K,+)$ se numește grupul aditiv al corpului, iar grupul $(K\setminus\{0\},*)$ se numește grupul multiplicativ al elementelor nenule ale corpului.

Dacă, în plus,înmulțirea este comutativă (echivalent spus în axioma 2 scriem „grup abelian"), atunci tripletul $(K,+,*)$ se numește **corp comutativ**.

Grupul elementelor inversabile ale unui corp K este $U(K)=K\setminus\{0\}$.

Mulțimea $\mathbb{Q}$, respectiv $\mathbb{R}$ a numerelor raționale,respectiv reale înzestrată cu operațiile de adunare și înmulțire are o structură de corp comutativ, numit **corpul numerelor raționale**, respectiv **corp numerelor reale**.

Inelul $\mathbb{Z}_p$ al claselor de resturi modulo „p" (p-prim) este corp comutativ.

Prof Matematică Gradul I
Ghiulserin Sali Liceul Teoretic "Callatis" Mangalia

Exercitiu propus pentru clasa a X-a:

Să se rezolve ecuația: $\left[lnx+\frac{1}{2}\right]+\left[lnx\cdot\frac{1}{2}\right]$ -2=lnx+1- $\left[(lnx+1)\cdot\frac{1}{2}\right]$.

Soluție: Ecuația se scrie

$$\left[y+\frac{1}{2}\right]+\left[y\cdot\frac{1}{2}\right]+\left[(y+1)\cdot\frac{1}{2}\right]=y+3$$

.

Folosind de două ori egalitatea $\left[x+\frac{1}{2}\right]+[x]=[2x]$

obținem ecuația

$[2y]=y+3$.

Notăm y+3=n, $n\in Z$ și atunci y=n-3. Ecuația devine:

$[2n-6]=n$

$n\leq 2n-6<n+1$

$6\leq n<7$

Cum n este întreg rezultă n=6 ,y=3 si $x=e^3$.

Prof MatematicĂ Gradul I
Bechir Ghiulnar Liceul Teoretic "Callatis" Mangalia

Primitive

În analiza matematică, o **primitivă** sau **integrală nedefinită** a unei funcții *f* este o funcție *F* a cărei derivate este egală cu *f*, adică, $F' = f$. Procesul de calcul al primitivelor se numește **primitivare** (sau **integrare nedefinită**). Primitivele sunt legate de integralele definite prin teorema fundamentală a calculului integral, și furnizează un mijloc convenabil de calcul al integralelor definite ale multor funcții.

Primitivele sunt importante deoarece pot fi utilizate la calculul integralelor definite, folosind teorema fundamentală a calculului integral: dacă *F* este o primitivă a unei funcții integrabile *f*, atunci:

$$\int_a^b f(x)\,dx = F(b) - F(a).$$

Din acest motiv, una din infinit de multele primitive ale unei funcții date *f* este uneori numită "integrală generală" sau "integrală nedefinită" a lui *f* și este scrisă folosind simbolul de integrală fără limite:

$$\int f(x)\,dx.$$

Dacă *F* este o primitivă a lui *f*, și *f* este definită pe un interval, orice altă primitvă *G* a lui *f* diferă de *F* printr-o constantă: există un număr *C* astfel încât *G*(*x*) = *F*(*x*) + *C* oricare ar fi *x*. *C* este numită constantă de integrare. Dacă domeniul lui *F* este o reuniune de două sau mai multe intervale disjuncte, atunci se pot alege constante de integrare diferite pentru fiecare interval. De exemplu :

$$F(x) = \begin{cases} -\frac{1}{x} + C_1 & x < 0 \\ -\frac{1}{x} + C_2 & x > 0 \end{cases}$$

este primitiva cea mai generală pentru $f(x) = 1/x^2$ pe domeniul său general $(-\infty, 0) \cup (0, \infty)$.

Calculul primitivelor funcțiilor elementare este adesea considerat mai dificil decât găsirea derivatelor acestora. Pentru unele funcții elementare, este imposibil să se exprime primitivele în termeni de alte funcții elementare.
Avem la dispoziție mai multe metode:

1.Liniaritatea integrării ne permite să descompunem integrale mai complexe în altele mai simple

2.Integrarea prin substituție, adesea combinată cu identitățile trigonometrice sau cu logaritmul natural

3.Integrarea prin părți pentru integrarea produsului de funcții

Prof Matematică Gradul I
Bechir Ghiulnar Liceul Teoretic "Callatis" Mangalia

MATEMATICĂ

Legi de compozitie

În mod frecvent se vorbește despre *operații* pe anumite mulțimi. De exemplu, operația de scădere a numerelor întregi este un procedeu prin care perechii de numere întregi $(x, y) \in \mathbb{Z} \times \mathbb{Z}$ i se asociază numărul întreg $x - y \in \mathbb{Z}$. Este important să se considere perechea sau mulțimea ordonată *(x,y)* și nu mulțimea *{x,y}*, deoarece contează ordinea în care apar *x* și *y*. De exemplu, perechii *(y,x)* îi corespunde prin această operație numărul $y - x \in \mathbb{Z}$, care în general diferă de *x*-*y*.

Generalizând, fie *M* o mulțime nevidă. Se numește **lege de compoziție internă** (sau **operație algebrică**) pe mulțimea *M* orice funcție definită pe *M* × *M* cu valori în *M*:

$$* : M \times M \to M, (x, y) \mapsto x * y$$

care asociază fiecărei perechi $(x, y) \in M \times M$ un element unic $x * y \in M$. Elementul $x * y \in M$ se citește **x compus cu y**.

O operație algebrică poate fi notată prin mai multe simboluri, de exemplu, $+, -, \times, \oplus, \circ, \odot, \cap, \cup, \Delta,$ etc.

Proprietățile unei operații

Fie o mulțime nevidă *M* și o operație * pe *M*. Spunem că:

1° Operația este **asociativă** dacă . $(x * y) * z = x * (y * z), \forall x, y, z \in M$

2° Operația * este **comutativă** dacă $x * y = y * x, \forall x, y \in M$

3° Operația * are **elementul neutru e** dacă $\exists e \in M$ astfel încât $x * e = e * x = x, \forall x \in M$.

4° Dacă operația * are elementul neutru $e \in M$, spunem că un element $x \in M$ este **simetrizabil** față de operația * dacă $\exists x' \in M$ astfel încât $x * x' = x' * x = e$ (x′ se numește **simetricul** lui x).

Elev Turcu Răzvan-Andrei Clasa a XII a B

Clasele de resturi modulo n

Fiind dat un numar natural n, nenul, pentru orice numar întreg k există numerele unice q (intreg) si r (natural, mai mic decat n), astfel încât a = nq + r.

Observații:

1) Numarul q este *câtul,* iar r este *restul împărțirii numărului a la n.*

2) Notație: r = a(mod n); se citește "*a modulo n*" si r se numește

redusul modulo n al numarului a.

3) Imaginandu-ne ca împărțim toate numerele întregi la n, este evident că resturile obținute sunt mai mari sau egale cu 0 (în cazul multiplilor lui n), dar mai mici, cel mult egale cu n - 1; deci există exact n tipuri de numere intregi, care se constituie în n submulțimi, disjuncte 2 cate 2, a căror reuniune formeaza mulțimea Z (se spune că se definește astfel o *partitie* a multimii numerelor intregi).

În cazul particular n = 3, se notează astfel:

$$\hat{0} = \{\ldots -10, -5, 0, 5, 10\ldots\} = \{5k | k \in \mathbb{Z}\},$$
$$\hat{1} = \{\ldots -9, -4, 1, 6, 11\ldots\} = \{5k+1 | k \in \mathbb{Z}\},$$
$$\hat{2} = \{\ldots -8, -3, 2, 7, 12\ldots\} = \{5k+2 | k \in \mathbb{Z}\},$$

4) In general, pentru un n oarecare, submulțimile respective sunt:

$$\hat{0},\ \hat{1},\ \hat{2},\ \cdots, \widehat{n-1},$$

ele continând toate numerele întregi de forma

nk, nk + 1, nk + 2, ... , respectiv nk + (n - 1),

unde k parcurge mulțimea Z.

Elev Turcu Răzvan-Andrei Clasa a XII a B

Coordonator revistã

Prof.Ghiulserin Sali

Colectiv redactional:
Prof.Pilat Elena-Mihaela
Prof.Ghiulnar Bechir
Informatician Bargan Gabriela
Elev Turcu Rãzvan Andrei clasa a XII a B

ISSN 2068-3642