***Ministerul Educaţiei,Tineretului si Sportului a Republicii Moldova***

### **Universitatea Tehnică a Moldovei**

***Facultatea Calculatoare, Informatică si Microelectronică***

Referat

Lucrarea de laborator Nr.2

la disciplina Programarea orientată pe obiecte

***Tema: Constructorul – funcţie de iniţializare a obiectelor clasei***

**Varianta 10**

## A efectuat

## studentul grupei TI-161

***Grosu Elena***

***A verificat***

***Prof. Univ. Botnaru S.***

**Tema:** Constructorul – funcţie de iniţializare a obiectelor clasei

**Scopul lucrării:**

* Studierea principiilor de definire şi utilizare a constructorilor
* Studierea principiilor de definire şi utilizare a destructorilor
* Studierea tipurilor de constructori

Varianta 10

а) Să se creeze clasa *Soft* – fişier, care conţine informaţia despre numele complet al fişierului şi anexa de asociere (doc – Word, psd – Photoshop, etc.), utilizînd memoria dinamică, mărimea, data şi timpul creării. Să se definească toti constructorii. Constructorul de schimbare a tipului are parametrul – numele de fişier. Să se definească funcţiile de redenumire a fişierului, de transfer în alt directoriu şi de modificare a anexei de asociere.

b) Să se creeze clasa *Matrix –* matrice. Clasa conţine pointer spre *long,* numărul de rînduri şi de coloane şi o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată şi constructorul cu doi parametri – matrice dreptunghiulară ş. a. Să se definească funcţiile membru de acces: returnarea şi setarea valorii elementului (i,j). Să se definească funcţiile de adunare şi scădere a două matrice; înmulţirea unei matrice cu alta; înmulţirea unei matrice cu un număr. Să se testeze funcţionarea clasei. În caz de insuficienţă de memorie, necorespondenţă a dimensiunilor matricelor, depăşire a limitei memoriei utilizate să se stabilească codul erorii.

**Întrebări de control:**

1. **Explicaţi termenul de iniţializare.**

crearea unui obiect de tipul clasei

1. **Ce cuvinte cheie se utilizează pentru definirea constructorului şi a destructorului?**

se utilizeaza cu acelasi nume ca al clasei , la destructor este precedat de “~”.

1. **Poate oare un constructor să returneze o valoare?**

nu , nici chiar void

1. **Pot oare să existe clase fără constructori?**

nu , daca programatoru-l nu de clara constructorul si constructorul de coipiere este creat automat de catre compilator.

1. **Cîţi constructori poate conţine o clasă? Dar destructori? De ce?**

Clasa poate sa contina un numar nenumarat de constructori , in dependenta de caz . Dar destructorul poate fi numai unul pentru ca el nu are paramaetri deci nu poate fi supraincarcat , si este unic pentru o clasa.

1. **Enumeraţi tipurile de constructori?**

Constructor fara parametri ( sau constructor implicit)

Constructor cu parametri

1. Constructor general (care initializeaza toate cimpurile)
2. Constructori cu un anumit numar de parametri

Constructor de copiere( este un constructor care creeaza un obiect null prin cloanrea unei instante deja existente)

Constructor de schimabare a tipului ( este un constructor care primeste un tip de date strain pentru clasa respectiva si il converteste in obiect de tipul dat )

1. **Cum se apelează un constructor?**

constructorul se noteaza ca o functie cu numele clasei cu sau fara parametri.

1. **De ce este nevoie de definit un constructor de copiere?**

Pentru ca putem avea scurgeri de memorie atunci cind alocam dinamic memoria si p/u ca constructorul generat automat implica copierea bit cu bit ceea ce nu intotdeauna este satisfacator.

1. **De ce este nevoie de definit un constructor de conversie a tipului?**

Pentru convertirea mai comoda a tipului de date.

1. **În ce cazuri se apelează constructorul de copiere? Care este sintaxa apelului?**

Constructorul de copiere se foloseste in urmatoarel cazuri:

* Initializarea unui obiect din altul de acelasi tip
* Copierea unui obiect p/u transmiterea lui ca a argument a unei functii
* Copierea unui obiect p/u returnarea lui din corpul unei functii

<numele clasei>(const <numele clsei> &<obiect>){ ... }

1. **Ce sunt listele de iniţializare (iniţializatori)?**

mecanism folosit in constuctori care contribuie la evitarea cheltuielii necesare pentru initializarea cimpurilor .

1. **În care caz se defineşte obligatoriu constructorul implicit?**

in cazul utilizării tabloului de obiecte se defineşte în mod obligatoriu un constructor implicit, care se apelează pentru iniţializarea elementelor tabloului.

**Listingul programului:**

**a)**

#include <iostream>

#include <iostream>

#include <cstring>

#include <stdio.h>

#include <process.h>

#include <stdlib.h>

#include <conio.h>

#include<ctime>

#include "windows.h"

using namespace std;

class soft

{

char \*nume;

char \*anexa;

public:

soft();

soft(char \*n,char \*a);

soft(char \*num);

soft(const soft &so);

~soft();

void redenum(char \*num)

{

delete[]this->nume;

nume=new char[strlen(num)+1];

nume=num;

}

void modif\_anexa(char\*anx)

{

delete[]this->anexa;

this->anexa=new char[strlen(anx)+1];

this->anexa=anx;

}

void afis()

{

cout<<"Numele fisierului: "<<this->nume<<endl;

cout<<"Anexa: "<<this->anexa<<endl;

}

};

soft::soft()

{

nume=new char[strlen("Fara nume")+1];

strcpy(nume,"Fara nume");

anexa=new char[strlen("Fara anexa")+1];

strcpy(anexa,"Fara anexa");

}

soft::soft(char \*n, char \*a)

{

nume=new char[strlen(n)+1];

strcpy(nume,n);

anexa=new char[strlen(a)+1];

strcpy(anexa,a);

}

soft::soft(char \*n)

{

nume=new char[strlen(n)+1];

strcpy(nume,n);

anexa=new char[strlen("Fara anexa")+1];

strcpy(anexa,"Fara anexa");

}

soft::soft(const soft &s)

{

nume=new char[strlen(s.nume)+1];

strcpy(nume,s.nume);

anexa=new char[strlen(s.anexa)+1];

strcpy(anexa,s.anexa);

}

soft::~soft()

{

if(nume) delete[]nume; nume=NULL;

if(anexa) delete[]anexa; anexa=NULL;

}

int main () {

int opt;

jump:

system("cls");

cout<<"1-Constructor implicit"<<endl;

cout<<"2-Constructor de copiere"<<endl;

cout<<"3-Constructor de schimbarea a tipului"<<endl;

cout<<"4-Redenumirea fisierului"<<endl;

cout<<"5-Modificarea anexei de asociere:"<<endl;

cout<<"6-EXIT:"<<endl;

cout<<"7-Time and date:"<<endl;

cin>>opt;

switch(opt)

{

case 1:

{

system("cls");

soft f1;

f1.afis();

getch(); goto jump;

}

case 2:

{

system("cls");

soft f1("FILE1",".pdf");

soft f2(f1);

f2.afis();

getch(); goto jump;

}

case 3:

{

system("cls");

char \*num=new char[12];

cout<<"Numele fisierului: "<<endl;

cin>>num;

soft f1(num);

f1.afis();

getch(); goto jump;

}

case 4:

{

system("cls");

soft f1("FILE1",".pdf");

f1.afis();

char \*num=new char[12];

cout<<"Numele nou al fisierului: "<<endl;

cin>>num;

f1.redenum(num);

f1.afis();

getch(); goto jump;

}

case 5:

{

system("cls");

soft f1("FILE1",".pdf");

f1.afis();

char \*num=new char[12];

cout<<"Anexa fisierului: "<<endl;

cin>>num;

f1.modif\_anexa(num);

f1.afis();

getch(); goto jump;

}

case 6:

{

system("cls");

exit(1);

}

case 7:

{

system("cls");

time\_t now=time(0);

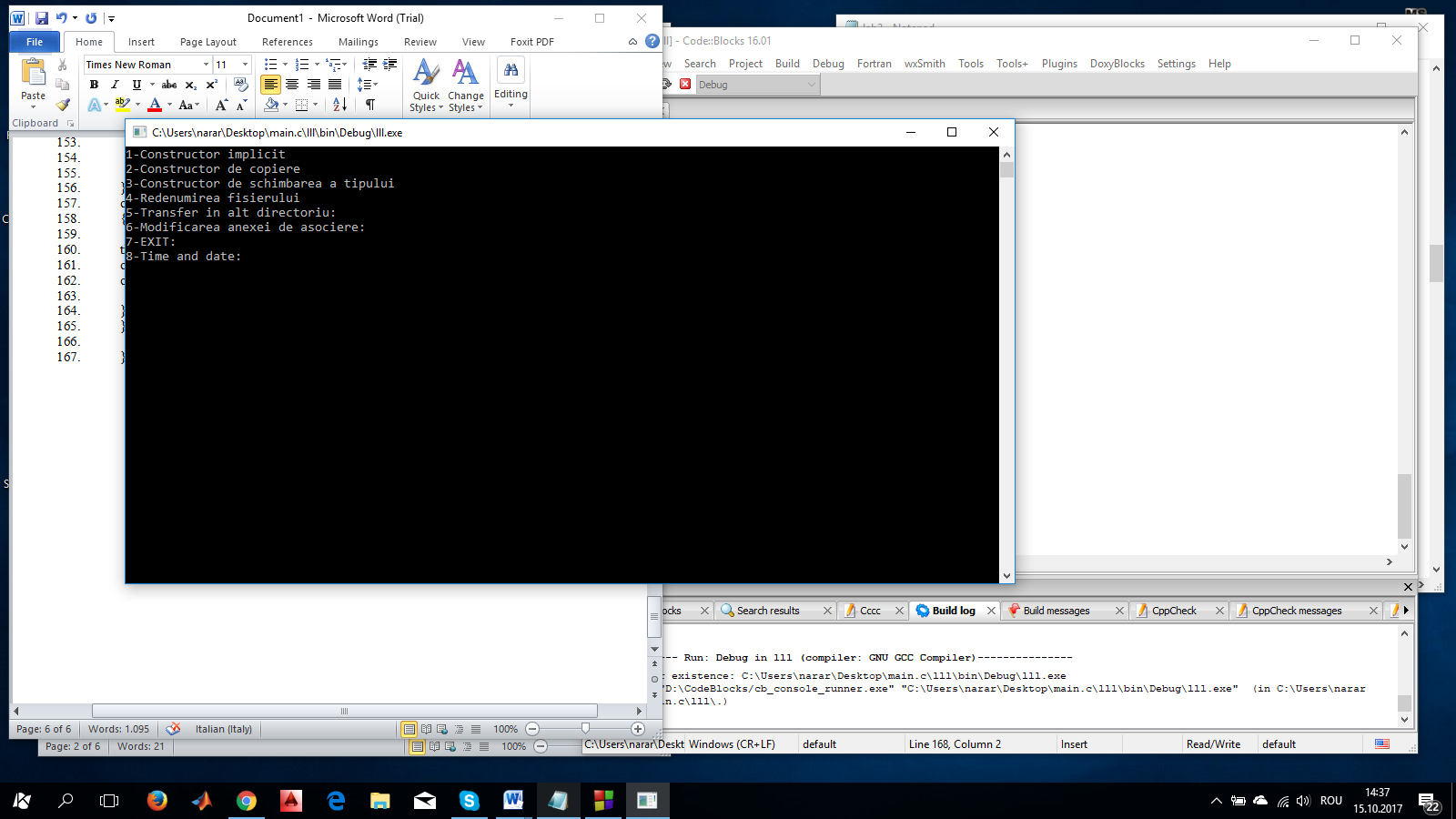
char \*dt=ctime(&now);

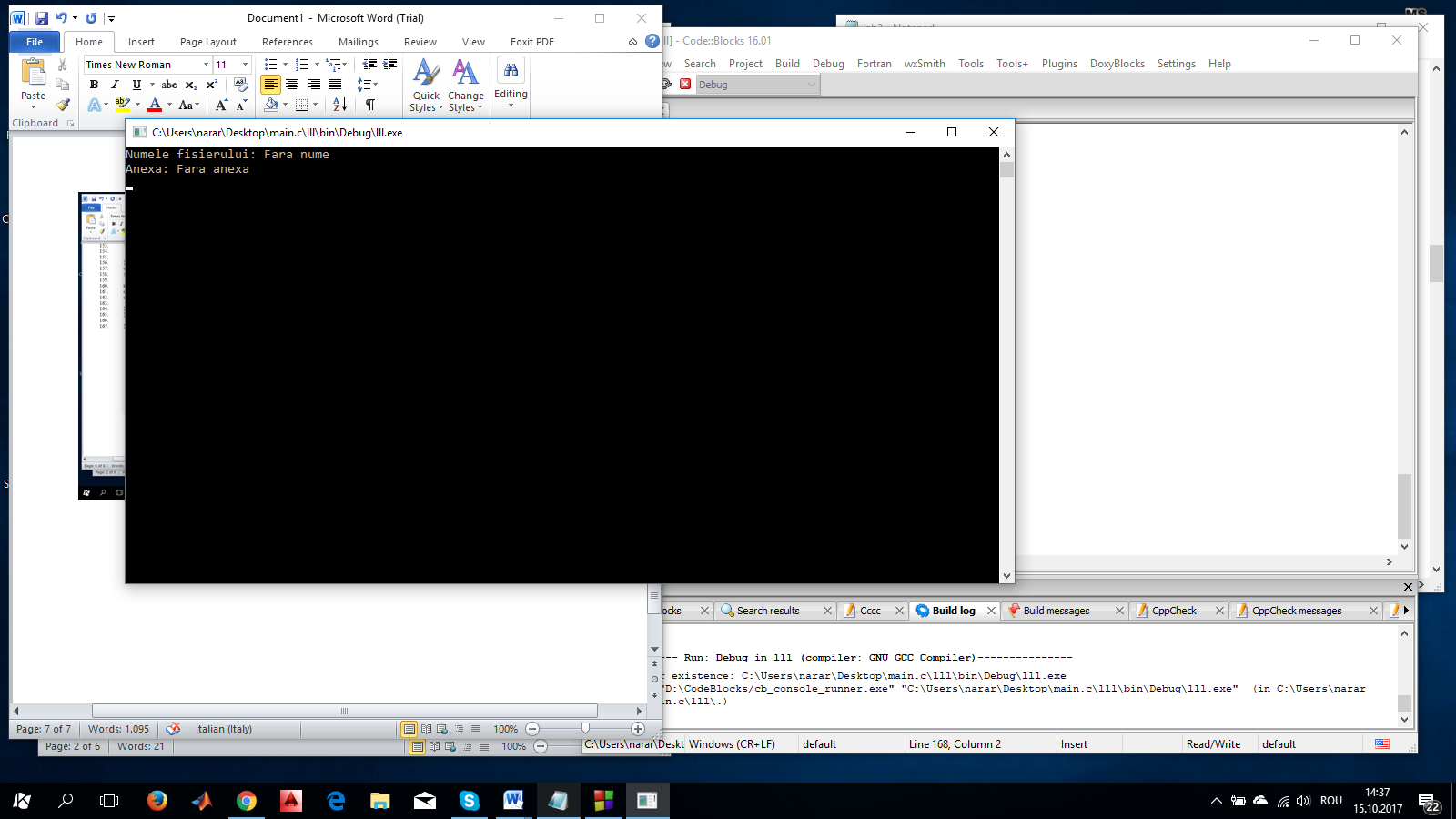
cout<<"Data si ora locala:"<<dt<<endl;

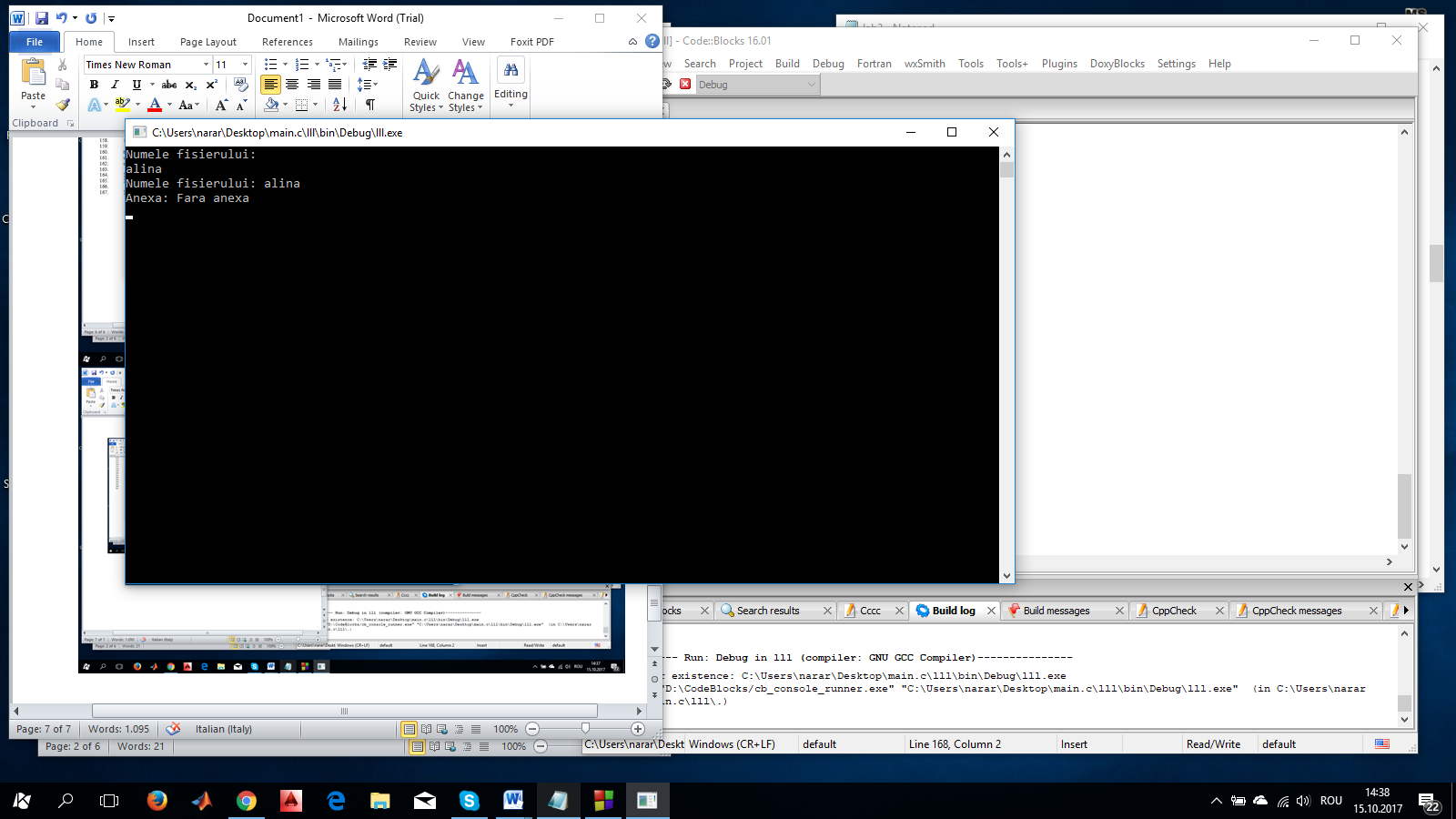
}

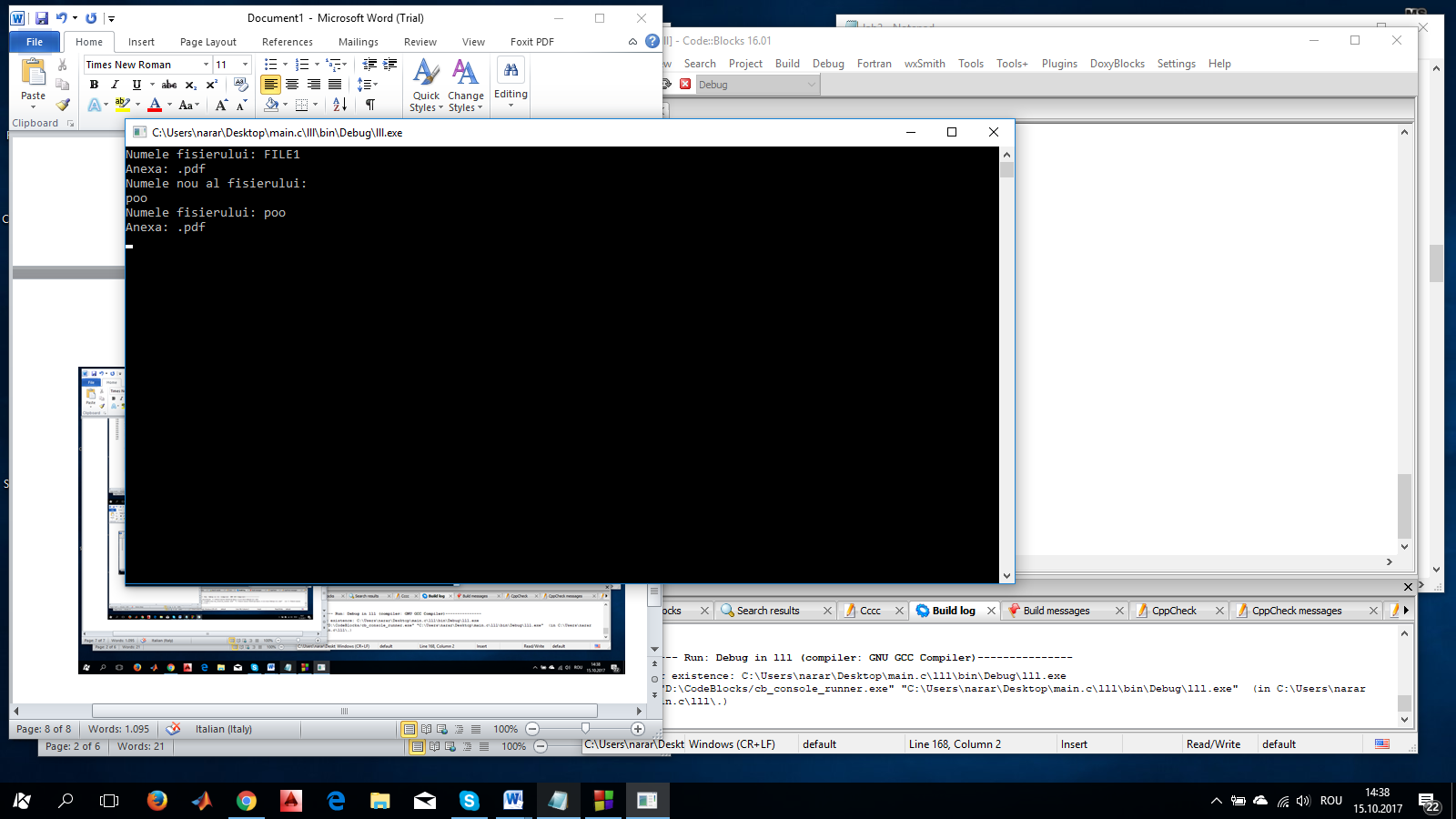
}

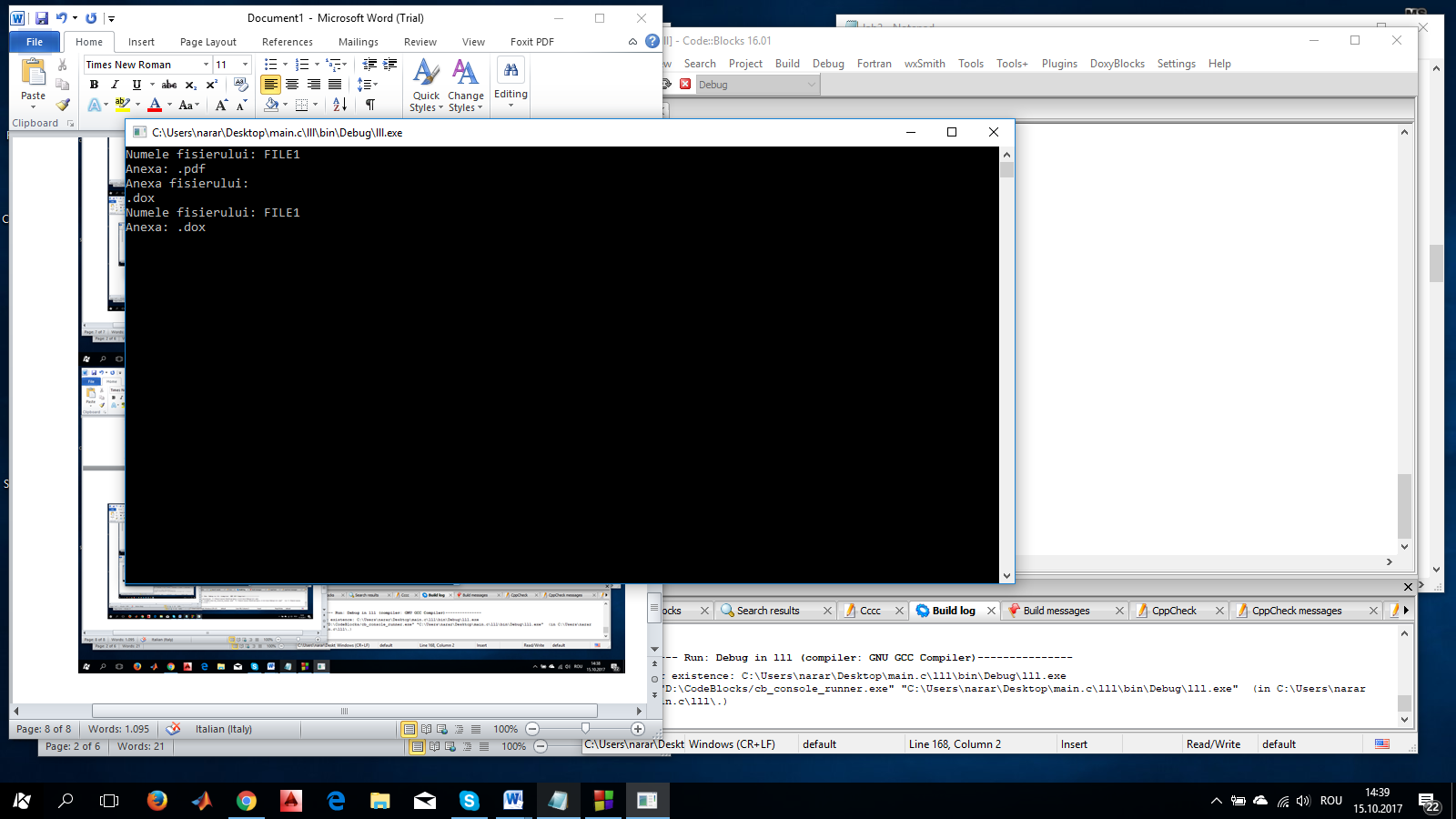
}

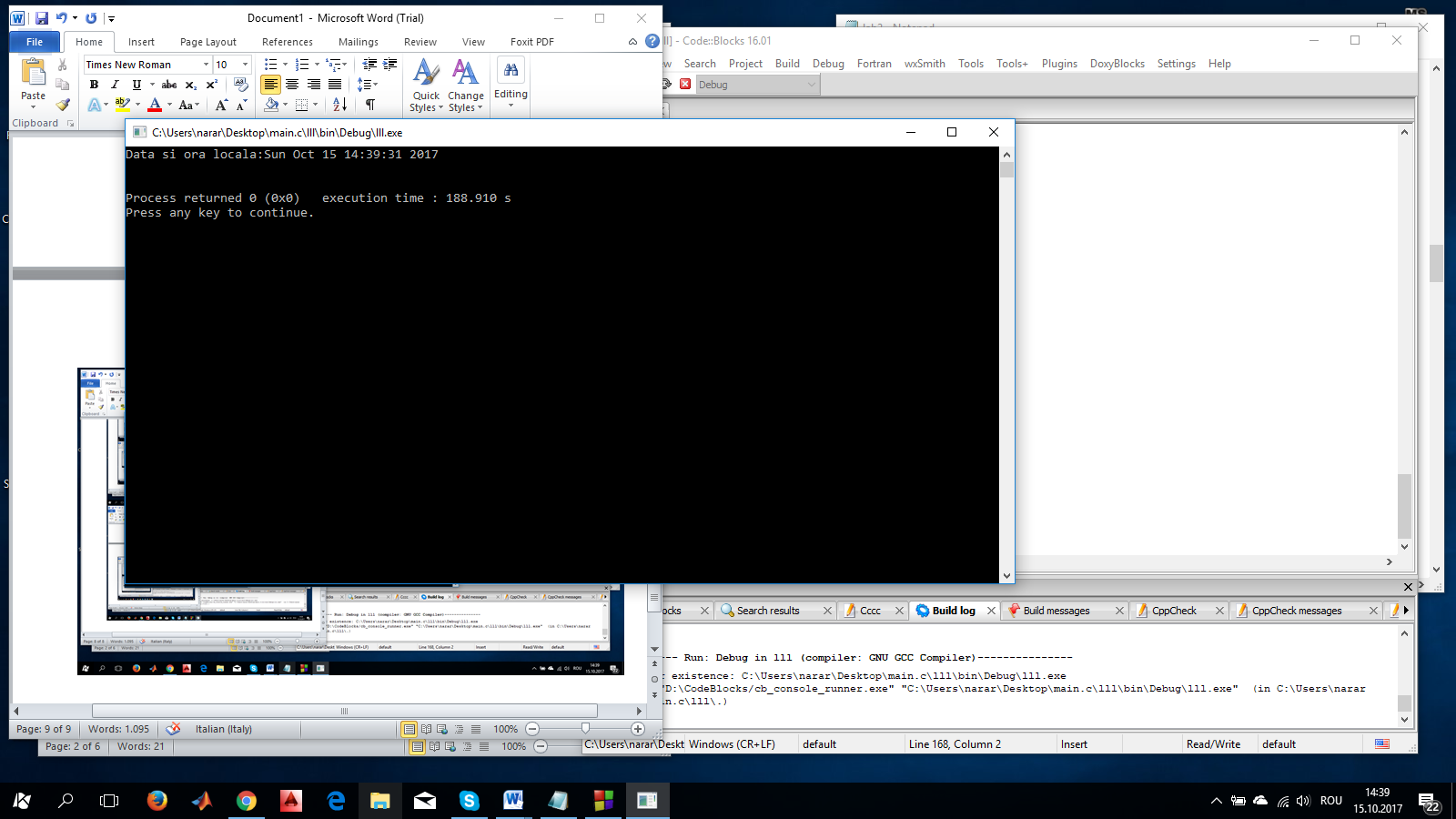












**b)**

#include <iostream>

#include <cstring>

#include <stdio.h>

#include <process.h>

#include <stdlib.h>

#include <conio.h>

using namespace std;

class matrix

{

long \*\*A;

int n, m, err;

public:

matrix();

matrix(int,int);

matrix(int);

matrix(const matrix&);

~matrix();

void intro();

void show();

void suma(const matrix &matr);

void produs(const matrix &matr);

void diferenta(const matrix &matr);

float get\_element(int,int);

void inmultire(long);

float returneaza(int x, int y);

void setare(int x, int y, float z);

void print\_err();

int get\_err();

};

matrix::matrix()

{

err=0;

n=m=1;

if (!(A=new long\*[n])) err=1;

else

if (!(A[0]=new long[m])) err=1;

else

{

A[n][m]=0;

}

}

matrix::matrix(int dim)

{

err=0;

long val;

if (dim>=0)

{

n=m=dim;

if (!(A=new long\*[n])) err=1;

else

for(int i=0; i<n; i++)

if (!(A[i]=new long[m])) err=1;

for(int i=0; i<n; i++)

for(int j=0; j<m; j++)

{

cout<<"a["<<i<<","<<j<<"]=";

cin>>val;

A[i][j]=val;

}

}

else err=2;

}

matrix::matrix(int n1, int m1)

{

n=n1;

m=m1;

long val;

err=0;

if ((n>=0)&&(m>=0))

{

if (!(A=new long\*[n])) err=1;

else

for(int i=0; i<n; i++)

if (!(A[i]=new long[m])) err=1;

for(int i=0; i<n; i++)

for(int j=0; j<m; j++)

{

cout<<"a["<<i<<","<<j<<"]=";

cin>>val;

A[i][j]=val;

}

} else err=2;

}

matrix::matrix(const matrix& B)

{

err=0;

n=B.n;

m=B.m;

if (!(A=new long\*[n])) err=1;

else

for(int i=0; i<n; i++)

if (!(A[i]=new long[m])) err=1;

for(int i=0; i<n; i++)

for(int j=0; j<m; j++)

A[i][j]=B.A[i][j];

}

matrix::~matrix()

{

err=0;

for(int i=0; i<n; i++)

delete[] A[i];

delete[] A;

A=NULL;

n=m=0;

}

int matrix::get\_err()

{

return err;

}

void matrix::print\_err()

{

system("cls"); switch (err)

{

case 0: { cout<<"Totul e OK."<<endl; break;}

case 1: { cout<<"Nu este de ajuns memorie."<<endl; break;}

case 2: { cout<<"Dimensiunea matricei trebuie sa fie pozitiva."<<endl; break;}

case 3: { cout<<"Trecerea peste limitele matricei."<<endl; break;}

case 4: { cout<<"Dimensiunile matricelor nu coincid."<<endl; break;}

case 5: { cout<<"Matricele nu sunt compatibile pentru inmultire."<<endl; break;}

}

cout<<"Press any key to continue..."<<endl;

getch();

}

void matrix:: show( )

{

for(int i=0; i<n; i++)

{

for(int j=0; j<m; j++)

cout<<A[i][j]<<" ";

cout<<endl;

}

}

void matrix::suma(const matrix &matr)

{

err=0;

if ((n==matr.n)&&(m==matr.m))

{

for(int i=0; i<matr.n; i++)

for(int j=0; j<matr.m; j++)

A[i][j]+=matr.A[i][j];

}

else err=4;

}

void matrix::diferenta(const matrix &matr)

{

err=0;

if ((n==matr.n)&&(m==matr.m))

{

for(int i=0; i<matr.n; i++)

for(int j=0; j<matr.m; j++)

A[i][j]-=matr.A[i][j];

}

else err=4;

}

void matrix::produs(const matrix &vec)

{

err=0;

matrix temp(vec);

if (m==vec.n)

{

for(int i=0; i<n; i++)

for(int j=0; j<m; j++)

{

float tmp=0;

for(int k=0; k<vec.n; k++)

tmp+=A[i][k]\*vec.A[k][j];

temp.A[i][j]=tmp;

}

for(int i=0; i<n; i++)

for(int j=0; j<m; j++)

A[i][j]=temp.A[i][j];

}

else err=5;

}

void matrix::inmultire( long val)

{

for(int i=0; i<n; i++)

for(int j=0; j<m; j++)

A[i][j]\*=val;

}

float matrix::returneaza(int x, int y)

{

return A[x][y];

}

void matrix::setare(int x, int y, float z)

{

A[x][y]=z;

}

int main()

{

int b;

float k;

system("cls");

cout<<"Dati prima matrice"<<endl;

matrix m1(3);

cout<<"Matricea introdusa: "<<endl;

m1.show();

cout<<endl<<"Dati a doua matrice"<<endl;

matrix m2(3,3);

cout<<"Matricea introdusa: "<<endl;

m2.show();

getch();

matrix md1(m1),md2(m2),md5(m1),md7(m1);

matrix md3(m1),md4(m2),md6(m2),md8(m2);

m1.suma(m2);

cout<<"Suma m1+m2: "<<endl;

m1.show();

getch();

md1.diferenta(md2);

cout<<"Diferentza m2-m1: "<<endl;

md1.show();

md3.inmultire(5);

cout<<"Produsul m1\*5: "<<endl;

md3.show();

getch();

cout<<"Produsul m1\*m2:"<<endl;

md5.produs(md6);

md5.show();

cout<<"Returneaza elementul din m1 de pe pozitia (3,3):";

cout<<endl;

md7.show();

k=md7.returneaza(3-1,3-1);

cout<<"Elementul="<<k;

cout<<endl<<endl;

cout<<"Seteaza in m2 elementul de pe pozitia (1,3)=100:";

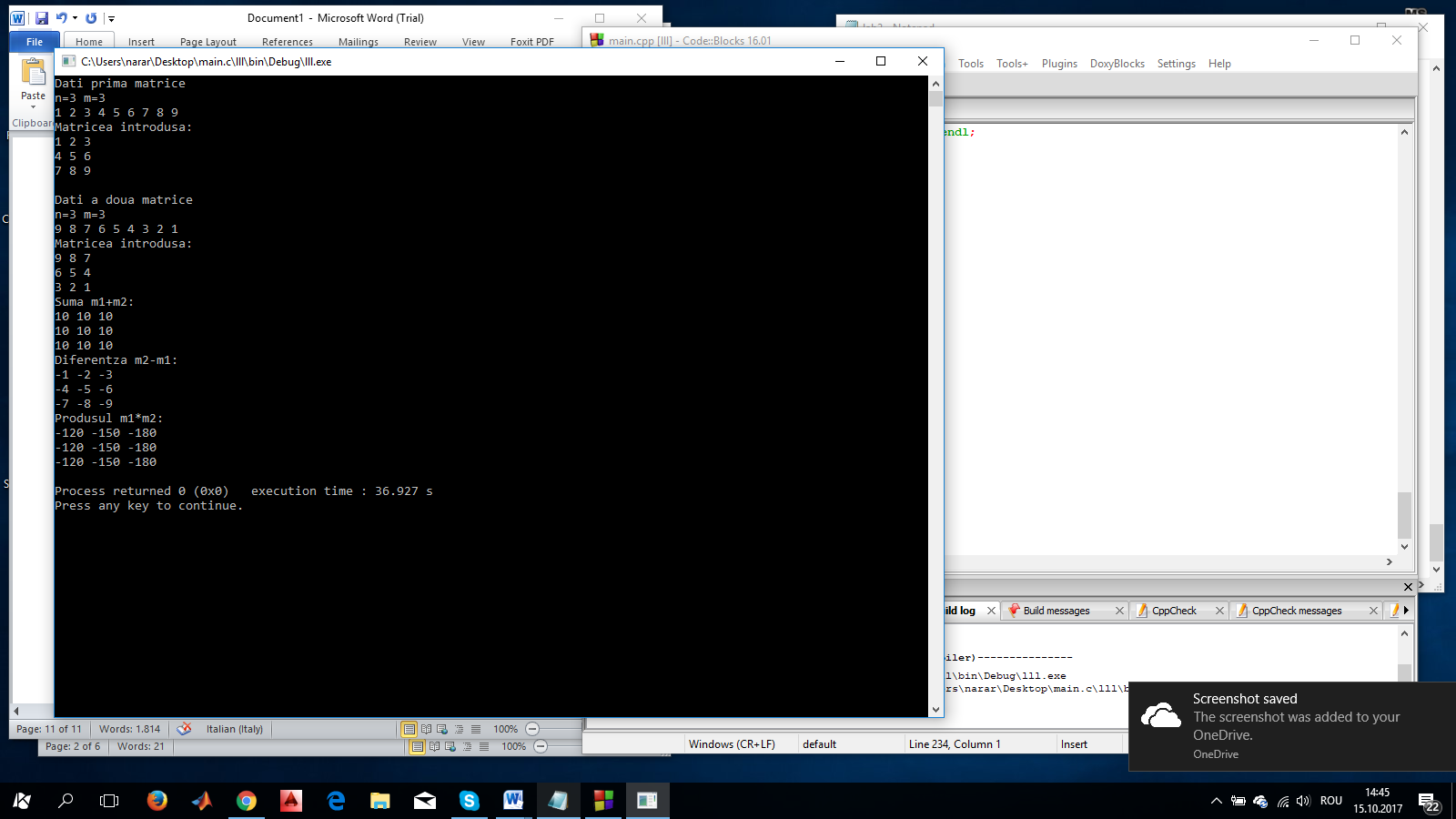
md8.setare(1-1,3-1,100);

cout<<endl;

md8.show();

getch();

return 0;

}

**Tema pentru acasa:**

1. Să se creeze clasa care conține următorii membri int first, int second. Să se efectueze supraîncărcarea operatorilor ”+”, “-”, “\*”, “/” , prin adunarea a două obiecte și salvarea rezultatelor în al treilea obiect.
2. Să se elaboreze un program carea ar permite supraîncărcarea operatorilor new si delete. Prin crearea clasei Student care conține: string name, int age. Să se implimenteze un constructor implicit și un constructor cu parametri.
3. Să se dezvolte un tip abstract de date pentru prelucrarea unei matrici bidimensionale, și realizează principalele operatii, prin operatori supraincarcati.

#include <iostream>

#include <cstring>

#include <stdio.h>

#include <process.h>

#include <stdlib.h>

#include <conio.h>

#include<ctime>

#include "windows.h"

using namespace std;

int i;

class numere{

public: int first;

int second;

public:

numere();

numere(int,int);

friend numere operator+(numere&,numere&);

friend numere operator\*(numere&,numere&);

friend numere operator-(numere&,numere&);

friend numere operator/(numere&,numere&);

void afis()

{

cout<<first;

cout<<second;

}

};

numere::numere()

{first=0; second=0;

}

numere::numere(int x,int y)

{first=x; second=y;

}

numere operator+(numere& y,numere& x)

{

numere w;

w.first=y.first+x.first;

w.second=y.second+x.second;

return w;

}

numere operator-(numere& y,numere& x)

{

numere w;

w.first=y.first-x.first;

w.second=y.second-x.second;

return w;

}

numere operator\*(numere& y,numere& x)

{

numere w;

w.first=y.first\*x.first;

w.second=y.second\*x.second;

return w;

}

numere operator/(numere& y,numere& x)

{

numere w;

w.first=y.first/x.first;

w.second=y.second/x.second;

return w;

}

int main()

{system("cls");

int x,opt,a,k;

jump:

system("cls");

cout<<"1-Suma a 2 nr"<<endl;

cout<<"2-Scaderea a 2 nr"<<endl;

cout<<"3-Produs a 2 nr"<<endl;

cout<<"4-Impartirea a 2 nr"<<endl;

cout<<"0-Exit"<<endl;

cin>>opt;

switch(opt)

{

case 1:

{system("cls");

cout<<"+"<<endl;

cout<<"Se citeste 1. int first= ";

cin>>a;

cout<<"Se citeste 1. int second= ";

cin>>k;

numere p(a,k);

cout<<"Se citeste 2. int first= ";

cin>>a;

cout<<"Se citeste 2. int second= ";

cin>>k;

numere n(a,k);

numere t;

t=p+n;

cout<<p.first<<"+"<<n.first<<"="<<t.first<<endl;

cout<<p.second<<"+"<<n.second<<"="<<t.second<<endl;

getch(); goto jump;}

case 2:

{system("cls");

cout<<"-"<<endl;

cout<<"Se citeste 1. int first= ";

cin>>a;

cout<<"Se citeste 1. int second= ";

cin>>k;

numere p(a,k);

cout<<"Se citeste 2. int first= ";

cin>>a;

cout<<"Se citeste 2. int second= ";

cin>>k;

numere n(a,k);

numere t;

t=p-n;

cout<<p.first<<"-"<<n.first<<"="<<t.first<<endl;

cout<<p.second<<"-"<<n.second<<"="<<t.second<<endl;

getch(); goto jump;}

case 3:

{system("cls");

cout<<"\*"<<endl;

cout<<"Se citeste 1. int first= ";

cin>>a;

cout<<"Se citeste 1. int second= ";

cin>>k;

numere p(a,k);

cout<<"Se citeste 2. int first= ";

cin>>a;

cout<<"Se citeste 2. int second= ";

cin>>k;

numere n(a,k);

numere t;

t=p\*n;

cout<<p.first<<"\*"<<n.first<<"="<<t.first<<endl;

cout<<p.second<<"\*"<<n.second<<"="<<t.second<<endl;

getch(); goto jump;}

case 4:

{system("cls");

cout<<"/"<<endl;

cout<<"Se citeste 1. int first= ";

cin>>a;

cout<<"Se citeste 1. int second= ";

cin>>k;

numere p(a,k);

cout<<"Se citeste 2. int first= ";

cin>>a;

cout<<"Se citeste 2. int second= ";

cin>>k;

numere n(a,k);

numere t;

t=p/n;

cout<<p.first<<"/"<<n.first<<"="<<t.first<<endl;

cout<<p.second<<"/"<<n.second<<"="<<t.second<<endl;

getch(); goto jump;}

case 0:

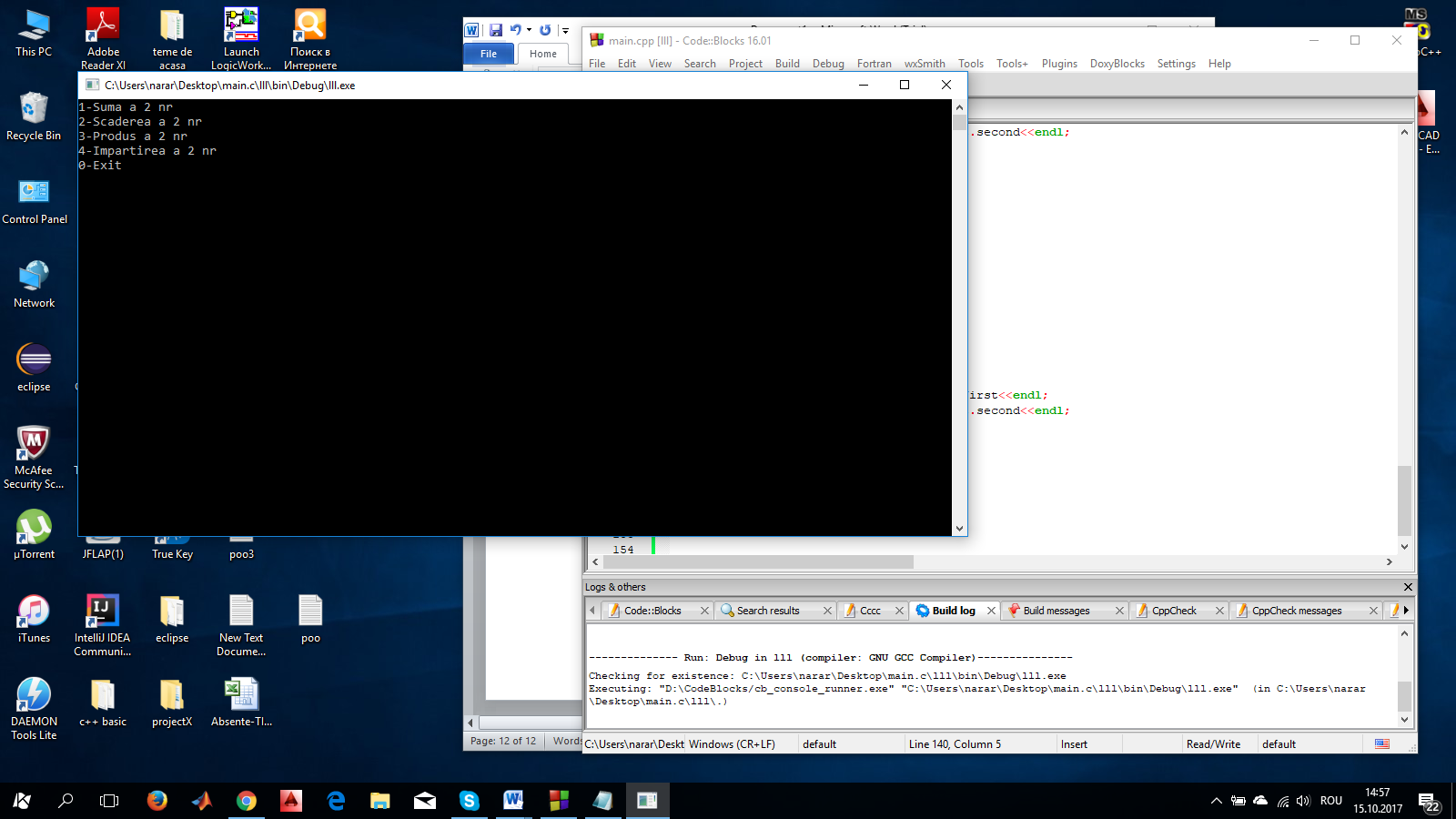
system("cls");

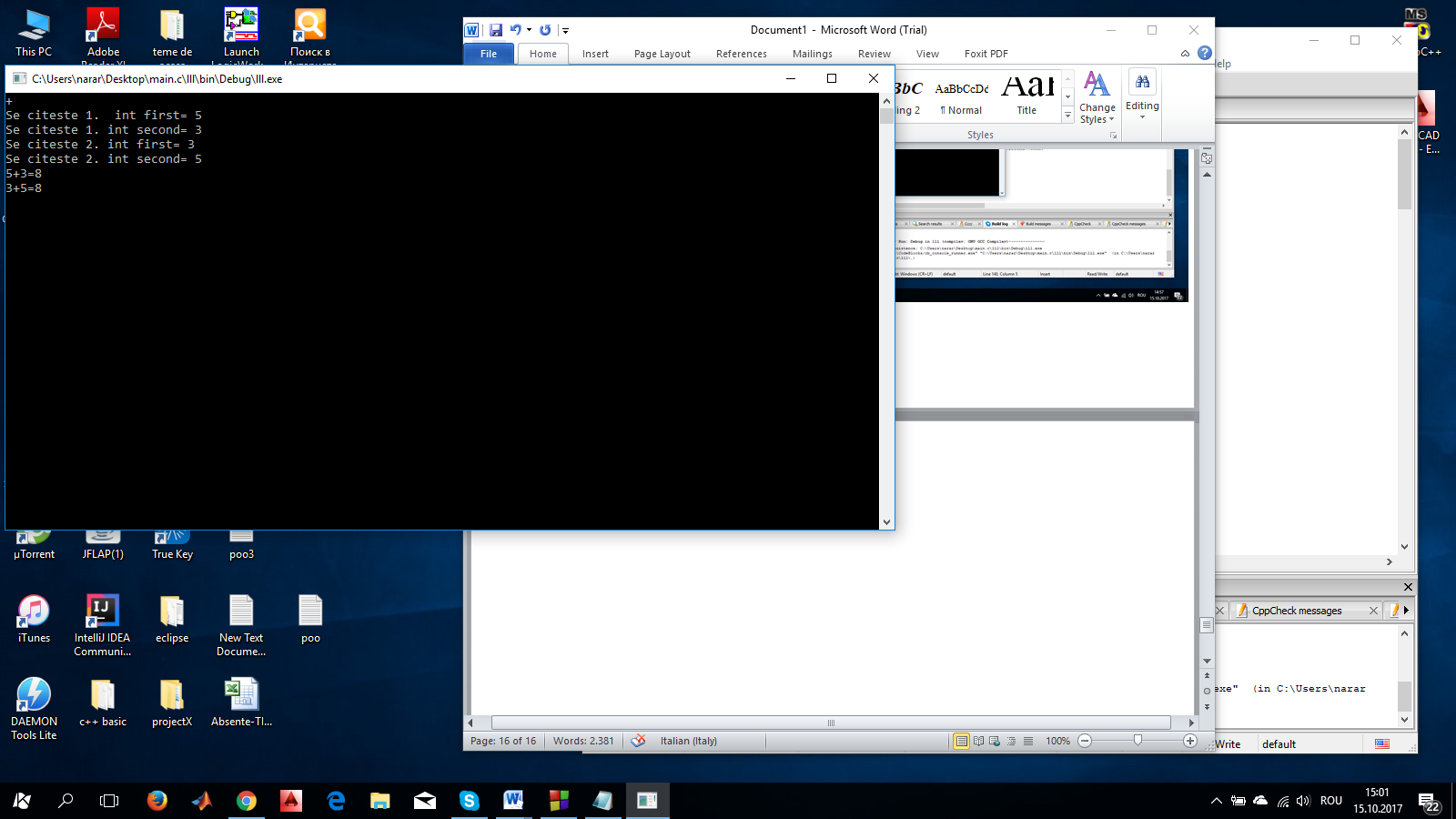
cout<<"Nu este disponibila functia";

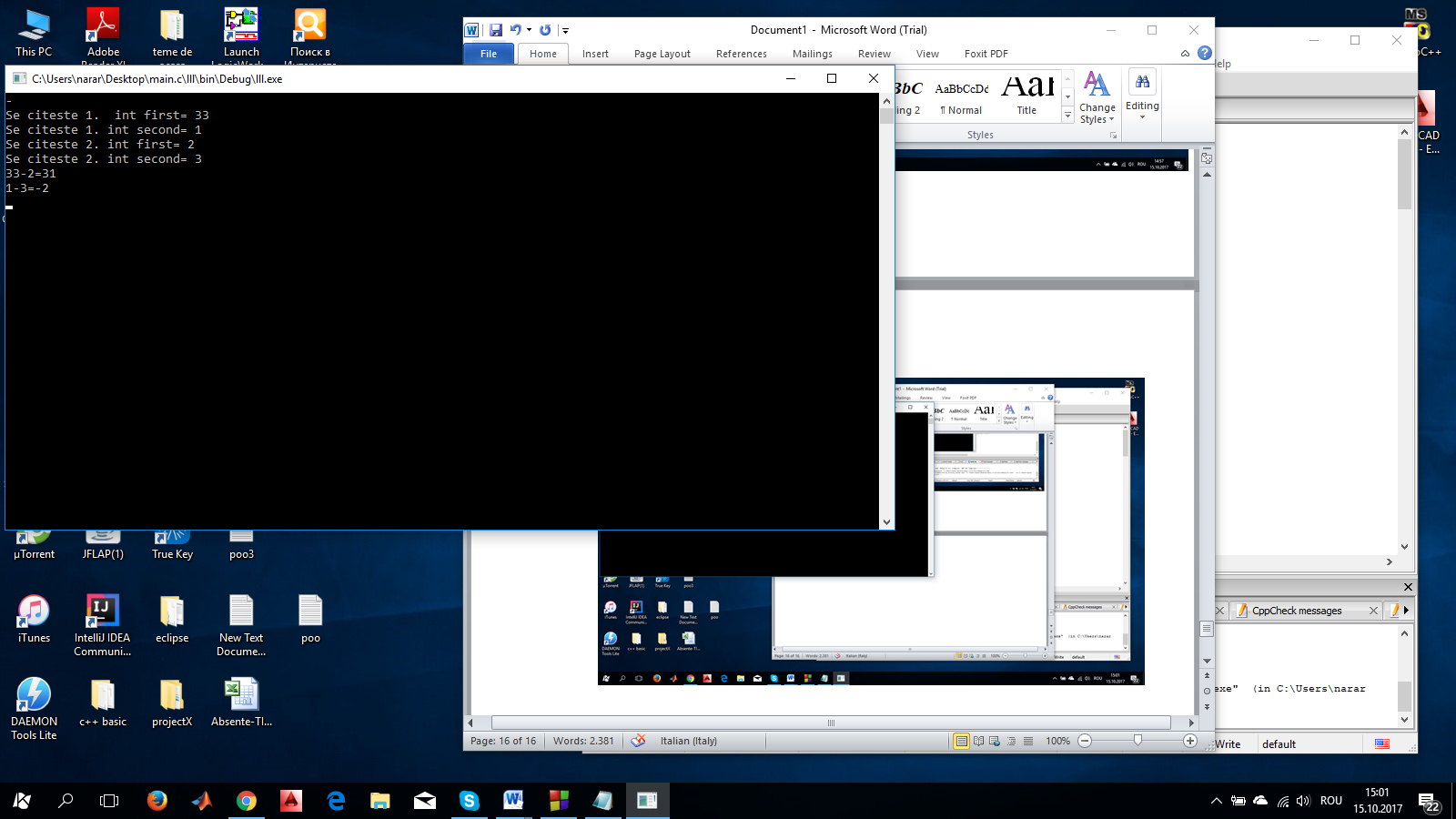
getch(); goto jump;

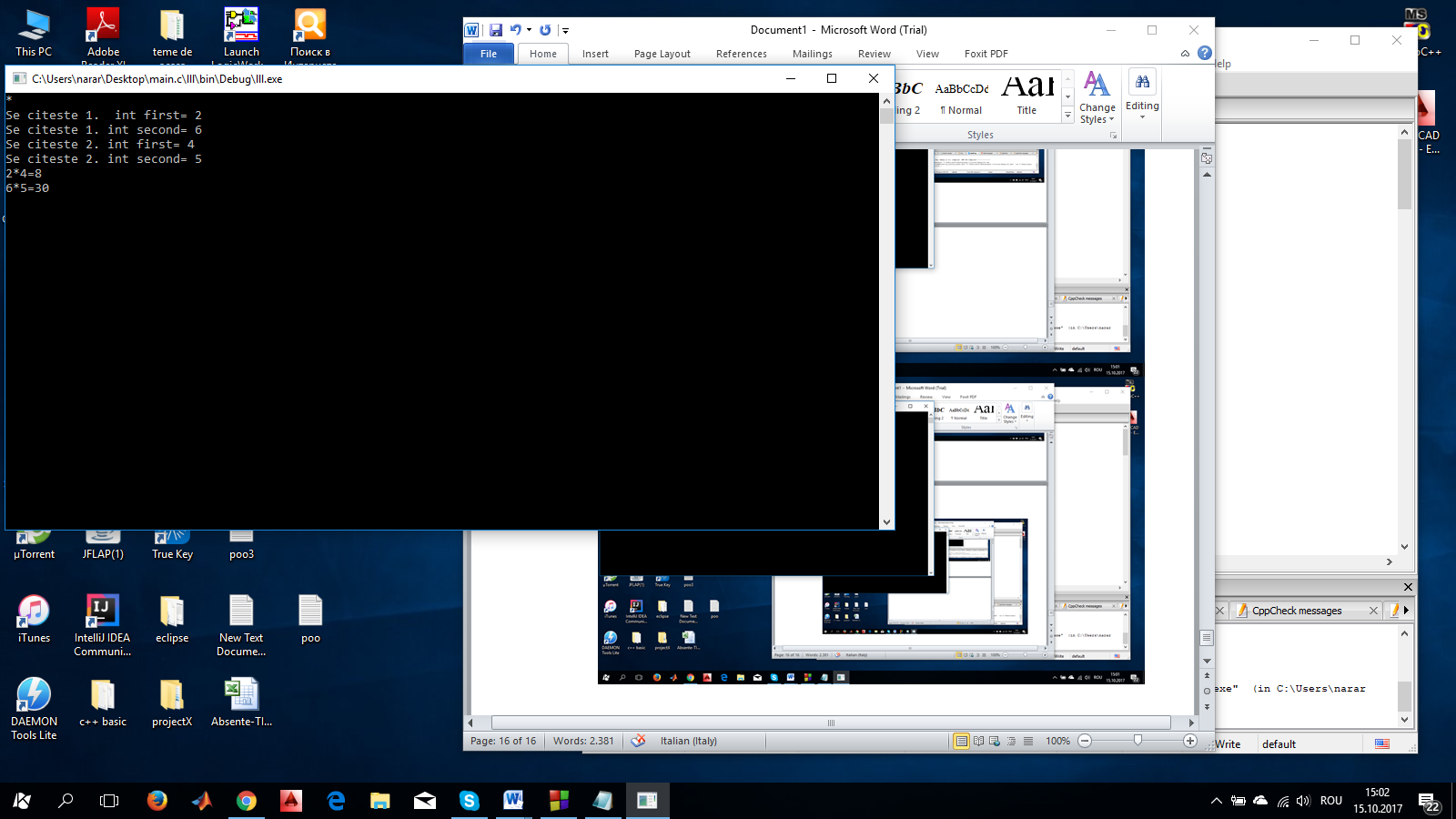
return 0;

}}









**2)**

#include<iostream>

#include<stdlib.h>

using namespace std;

class student

{

string name;

int age;

public:

student()

{

cout<< "Constructor este apelat\n" ;

}

student(string name, int age)

{

this->name = name;

this->age = age;

}

void show()

{

cout<< "Name:" << name << endl;

cout<< "Age:" << age << endl;

}

void \* operator new(size\_t size)

{

cout<< "Supraincarcarea operatorilor new cu dimensiunea : " << size << endl;

void \* p = ::new student();

//variant alternativa

//void \* p = malloc(size);

return p;

}

void operator delete(void \* p)

{

cout<< "Supraincarcarea operatorului delete " << endl;

free(p);

}

};

int main()

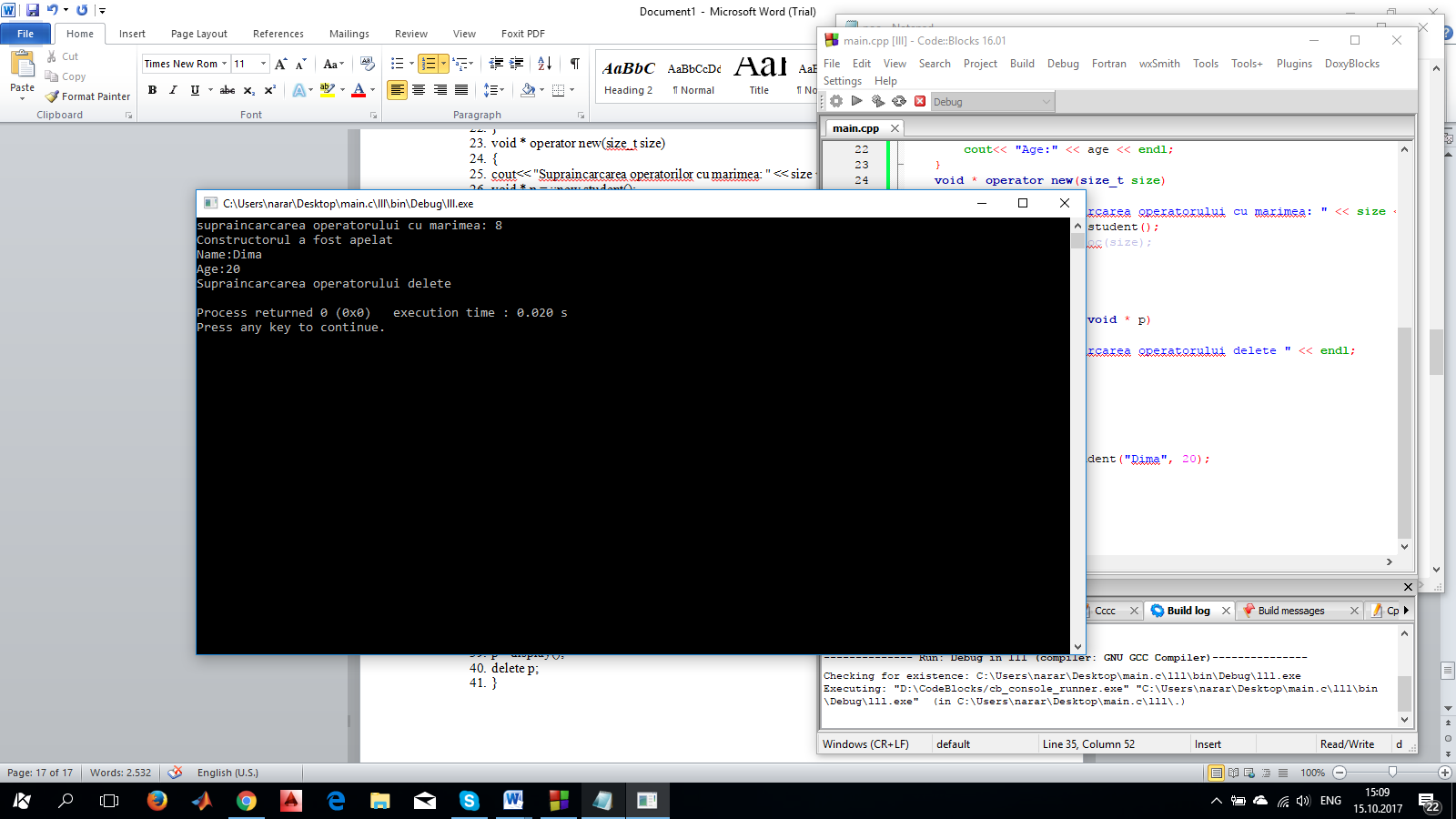
{

student \* p = new student("Dima", 20);

p->show();

delete p;

}



**3)**

#include <iostream>

#include <cstring>

#include <stdio.h>

#include <process.h>

#include <stdlib.h>

#include <conio.h>

#include<ctime>

#include "windows.h"

using namespace std;

class Mat {

public:

int \*val;

int total;

int row;

int col;

public:

Mat();

Mat(int,int);

Mat(const Mat& rhs);

Mat& operator=(const Mat& rhs);

Mat& operator+=(int scalar);

Mat& operator-=(int scalar);

Mat& operator\*=(int scalar);

void show()

{

for (int i=0;i<row;i++)

{

string del="";

for(int j=0;j<col;j++){

cout<<del<<val[i\*col+j];

del=",";

}

cout<<endl;

}

}

};

Mat::Mat()

{

val=new int [16];

total=16;

row=4;

col=4;

for (int i=0;i<total;i++)

{

val[i]=0;

}

}

Mat::Mat(int x, int y)

{

total=x\*y;

val=new int [x\*y];

row=x;

col=y;

for (int i=0;i<total;i++)

{

cin>>val[i];

}

}

Mat::Mat(const Mat& rhs)

{

total=rhs.total;

val=new int [rhs.total];

row=rhs.row;

col=rhs.col;

std::memcpy(val,rhs.val,rhs.total\*sizeof(int));

}

Mat& Mat::operator+=(int w)

{

for(int i=0;i<total;i++)

val[i]+=w;

return \*this;

}

Mat& Mat::operator-=(int w)

{

for(int i=0;i<total;i++)

val[i]-=w;

return \*this;

}

Mat& Mat::operator\*=(int w)

{

for(int i=0;i<total;i++)

val[i]\*=w;

return \*this;

}

int main()

{

int k,s;

cout<<"Nr. de rinduri:";

cin>>k;

cout<<"Nr. de coloane:";

cin>>s;

cout<<endl;

Mat a(k,s);

a.show();

cout<<endl;

cout<<"Matricea copiata:";

cout<<endl;

Mat b(a),c(a);

b.show();

cout<<endl;

int d;

cout<<"Suma cu scalarul: ";

cin>>d;

cout<<endl;

a+=d;

a.show();

cout<<endl;

cout<<"Diferenta cu scalarul: ";

cin>>d;

cout<<endl;

b-=d;

b.show();

cout<<"Produsul cu scalarul: ";

cin>>d;

cout<<endl;

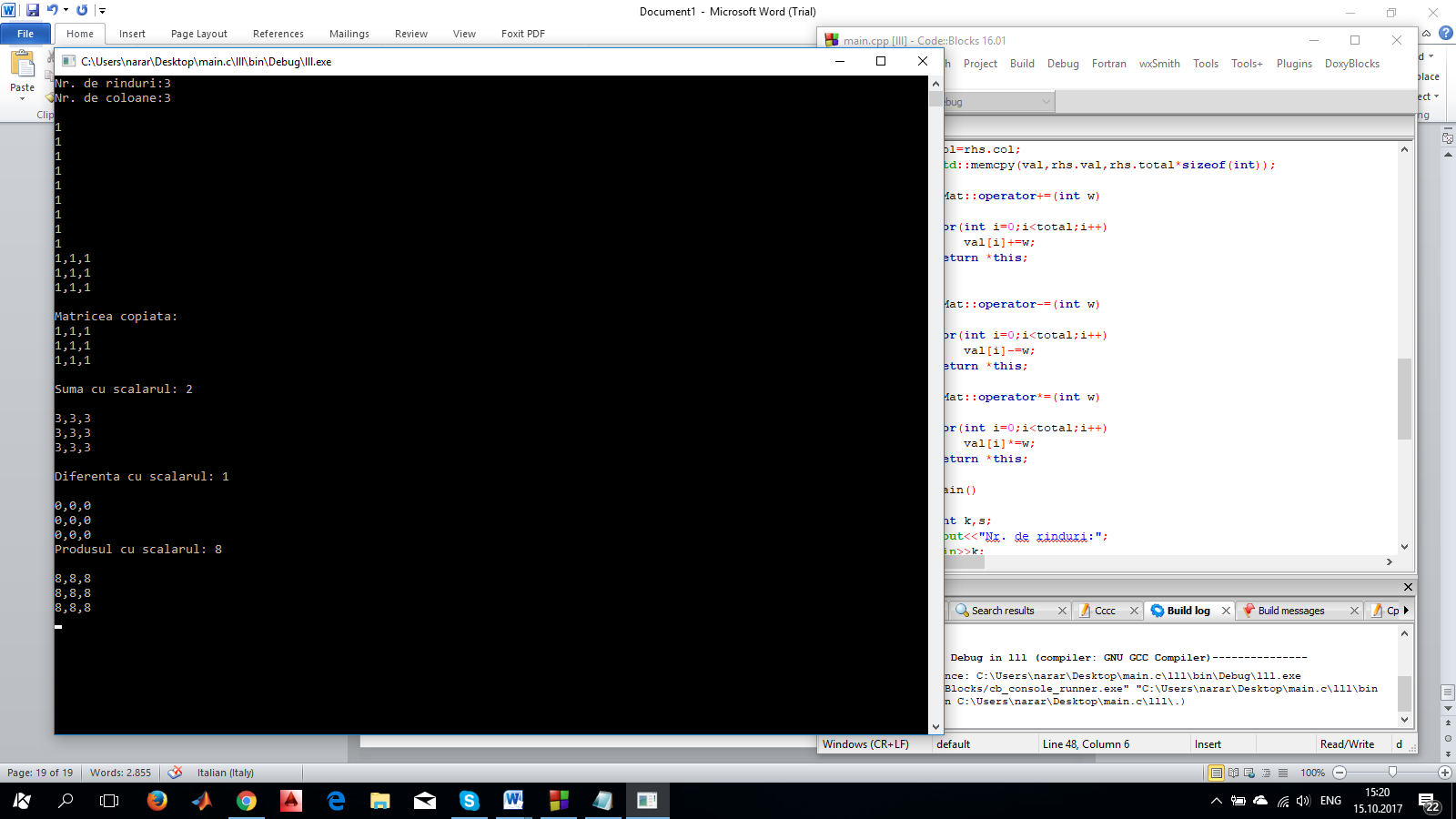
c\*=d;

c.show();

getch();

return 0;

}



**Concluzie:**

In procesul elaborarii lucrarii de laborator nr-2 s-a concretizat notiunea de clasa si de obiect care sta la baza programarii orientate pe obiect.Astfel s-a insusit deprinderea de a opera cu clasele care reprezinta o extindere a notiunii de structura deoarece clasa contine si functii membre numite metode.

Un rol de baza il semnifica si proprietatea de incapsulare, adica pastrarea integritatii datelor,componentelor obiectului.