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

### **Universitatea Tehnică a Moldovei**

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

Referat

Lucrarea de laborator Nr.3

la disciplina Programarea orientată pe obiecte

***Tema:* Supraîncărcarea operatorilor**

**Varianta 10**

## A efectuat

## studentul grupei TI-161

***Grosu Elena***

***A verificat***

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

**Tema:** Supraîncărcarea operatorilor

**Scopul lucrării:**

* Studierea necesităţii supraîncărcării operatorilor;
* Studierea sintaxei de definire a operatorilor;
* Studierea tipurilor de operatori;
* Studierea formelor de supraîncărcare;

**Întrebări de control:**

1. **Este oare supraîncărcarea operatorilor absolut necesară?**

Da , supraincararea operatorilor e importanta, astfel noi putem invata operatorii de a lucra cu tipuri abstracte ca si cum ar fi tipuri de baza.

1. **Ce cuvinte cheie se utilizează pentru definirea operatorilor?**

Numele clasei la care redefinim operatorul , tip rezultatului daca il returneza , cuv. Cheie operator si insasi operatorul.

1. **Pot oare operatorii să nu întoarcă niciun rezultat?**

Da, operatorii pot returna rezultat

1. **Cum se clasifică operatorii?**

* Aritmetici
* Relationali
* Logici
* Pe biti

Dupa numar(unar , binari , ternari) In cazul supraincarcarii exista operatori care se supraincarca si operatori care nu se supraincarca.

1. **Cum deosebeşte compilatorul forma prefixă şi cea postfixă a operatorilor unari?**

Pentru a distinge postfixarea de prefixarea dor ii transmitem ca parametru cuv cheie int .

*Complex operator++(); // preincrement*

*Complex operator++(int); // postincrement*

1. **Care operatori nu pot fi supraîncărcaţi?**

‘#’ , ‘::’ , ’;’ , ‘;’ , ‘.’, ‘{}’ ,sizeof

1. **Care este sintaxa de apel a operatorilor?**

Sintaxa ramine aceeasi ca si la tipurile predefinite , atunci cind utilizam operatorul pentru tipul abstract compilatorul utilizeaza operatorul redefinit de noi.

1. **Cum se utilizează operatorul "()"?**

Operatorul se foloseste in mod diferit si la prima vedere poate fi confundat cu constructorul clasei.

1. **Cum se utilizează operatorul "->"?**

Operatorul dat este folosit p/u indicare parametrilor clasei.

1. **De ce sunt necesare două forme de supraîncărcare (ca funcţii membru şi ca funcţii prieten)?**

Se utilizea 2 tipuri de supraincarcare p/u ca operatorul care dorim sa-l predefinim deja apartine unei clase de sistem (ex operatori de intrare si iesire aprtin clasei istream si respecti ostream) de aceea sunt utlizate ca functii prietene.

1. **Cărui tip aparţine operatorul "<<" pentru ieşiri de obiecte?**

Operatorul dat apatine la tipul operatorilr de intrare/iesire care ase supraincarca su ajutorul functiilor pritetene .

1. **În care caz este necesar să se definească operatorul de atribuire?**

Cind alocam dinamic memoria.

Sarcina:

Varianta 10

а) Să se creeze clasa *Bool –* variabile logice. Să se definească operatorii "+" – SAU logic, "\*" – ŞI logic, "^" – SAU EXCLUSIV, ca funcţii prietene, iar operatorii "==" şi "!=" – ca metode ale clasei. Operatorii trebuie să permită realizarea operaţiilor atît cu variabilele clasei date, cît şi cu variabilele de tip predefinit *int*. (Dacă numărul întreg este diferit de zero, se consideră că variabila este adevăr, altfel – fals.)

b) Să se creeze o clasă *Set* – mulţimea numerelor întregi, utilizînd memoria dinamică. Să se definească operatorii de lucru cu mulţimile: "+" – uniunea, "\*" – intersecţia, "-" scăderea, ca metode ale clasei, iar "+=" – înserarea unui nou element în mulţime, "==" – comparare la egalitate, ş. a. ca funcţii prietene. Să se definească operatorii "<<" şi ">>". Să se definească funcţia de verificare a apartenenţei unui element la o mulţime.

***Listingul programului:***

***a)***

#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 boo1{

int b;

public:

boo1();

boo1(int);

friend int operator+(boo1&,boo1&);

friend int operator+(int&,boo1&);

friend int operator\*(boo1&,boo1&);

friend int operator\*(int&,boo1&);

friend int operator-(boo1&,boo1&);

friend int operator-(int&,boo1&);

int operator==(boo1&);

int operator==(int&);

int operator!=(boo1&);

int operator!=(int&);

int afis()

{

return b;

}

};

boo1::boo1()

{b=0;

}

boo1::boo1(int x)

{if(x)b=1;else b=0;

}

int boo1::operator==(boo1& x)

{

if(b==x.b) return 1;

else return 0;

}

int boo1:: operator==(int& x)

{

if(b==x) return 1;

else return 0;

}

int boo1:: operator!=(boo1& x)

{

if(b!=x.b) return 1;

else return 0;

}

int boo1:: operator!=(int& x)

{

if(b!=x) return 1;

else return 0;

}

int operator+(boo1& y,boo1& x)

{

if(y.b||x.b)return 1;

else return 0;

}

int operator+(int& y,boo1& x)

{

if(y||x.b)return 1;

else return 0;

}

int operator\*(boo1& y,boo1& x)

{

if(y.b==0||x.b==0)return 0;

else return 1;

}

int operator\*(int& y,boo1& x)

{

if(y==0||x.b==0)return 0;

else return 1;

}

int operator-(boo1& x,boo1& y)

{

if(y.b!=x.b)return 1;

else return 0;

}

int operator-(int& x,boo1& y)

{

if(y.b!=x)return 1;

else return 0;

}

int main()

{system("cls");

int x,opt,a,k;

jump:

system("cls");

cout<<"1-bool+bool"<<endl;

cout<<"2-bool+int"<<endl;

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

cin>>opt;

switch(opt)

{

case 1:

{system("cls");

cout<<"bool bool"<<endl;

cout<<"Se citeste variabila logica p= ";

cin>>a;

cout<<"Se citeste variabila logica n= ";

cin>>k;

boo1 p(a),n(k);

cout<<p.afis()<<"+"<<n.afis()<<"="<<p+n<<endl;

cout<<p.afis()<<"\*"<<n.afis()<<"="<<p\*n<<endl;

cout<<p.afis()<<"^"<<n.afis()<<"="<<p-n<<endl;

getch(); goto jump;}

case 2:

{system("cls");

cout<<"bool int"<<endl;

cout<<"Se citeste variabila logica p= ";

cin>>a;

cout<<"Se citeste variabila integer n= ";

cin>>x;

if(x==0) x=0;

else x=1;

boo1 p(a);

cout<<p.afis()<<"+"<<x<<"="<<x+p<<endl;

cout<<p.afis()<<"\*"<<x<<"="<<x\*p<<endl;

cout<<p.afis()<<"^"<<x<<"="<<x-p<<endl;

getch(); goto jump;}

case 0:

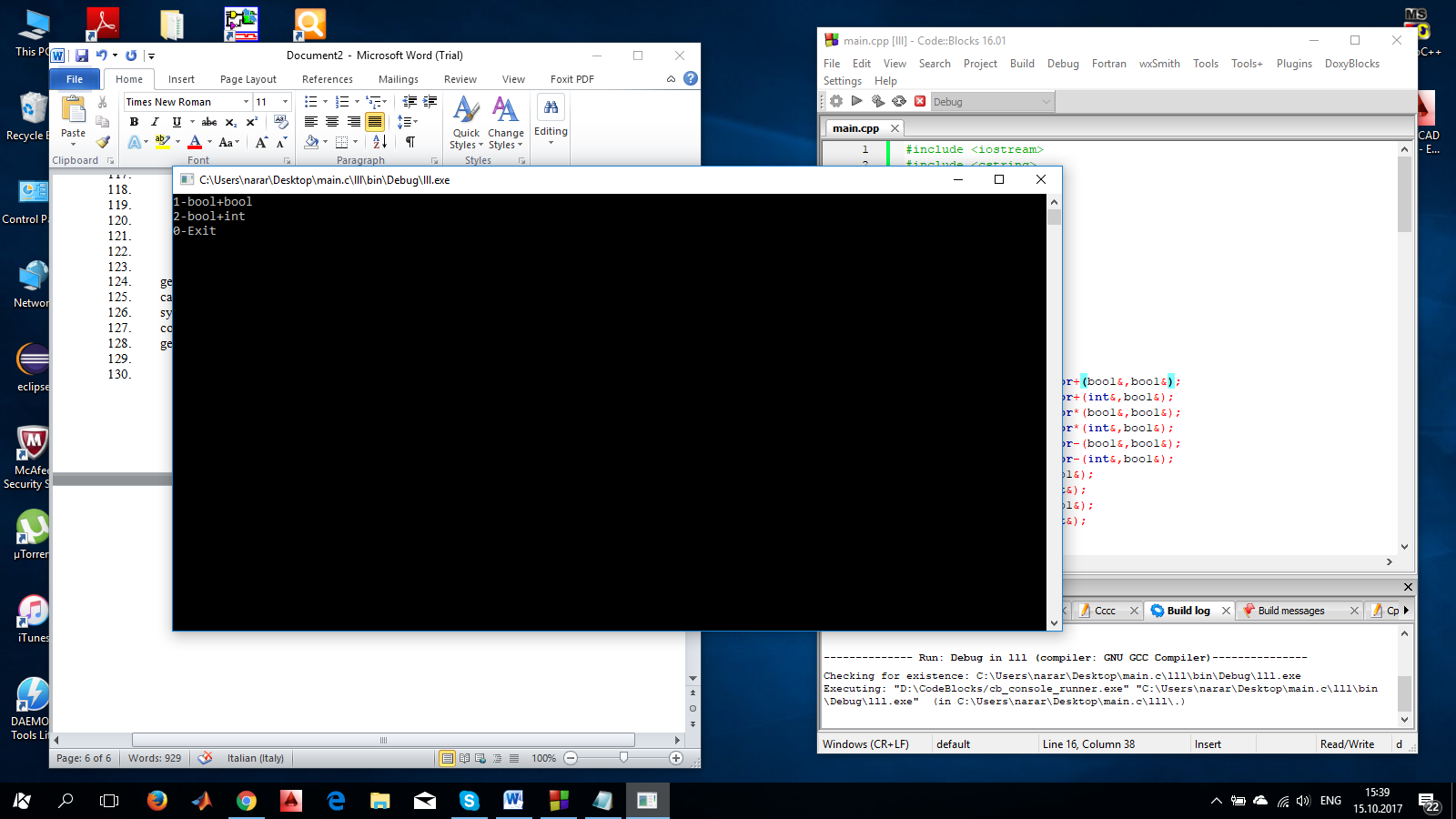
system("cls");

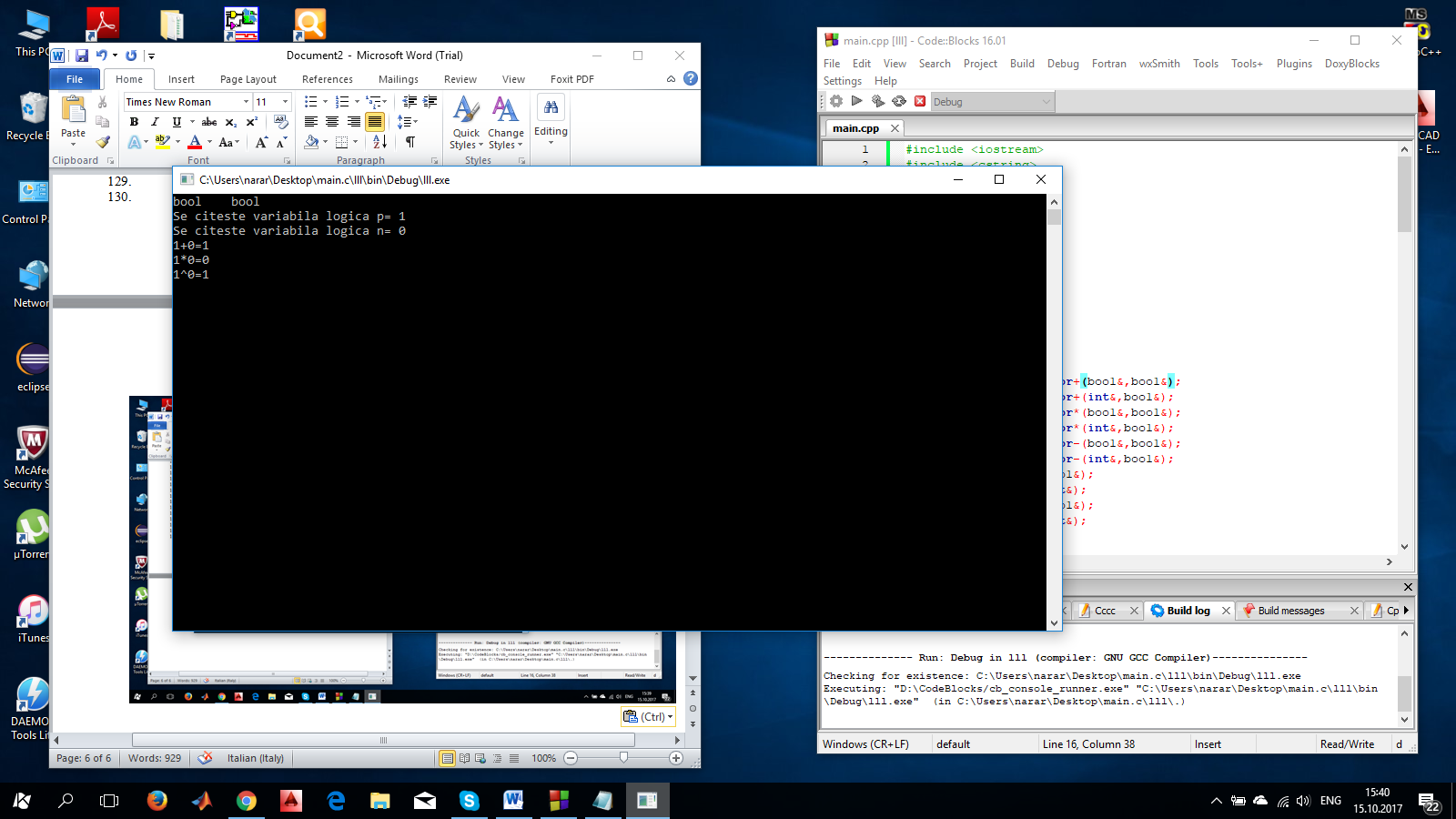
cout<<"Nu este disponibila functia";

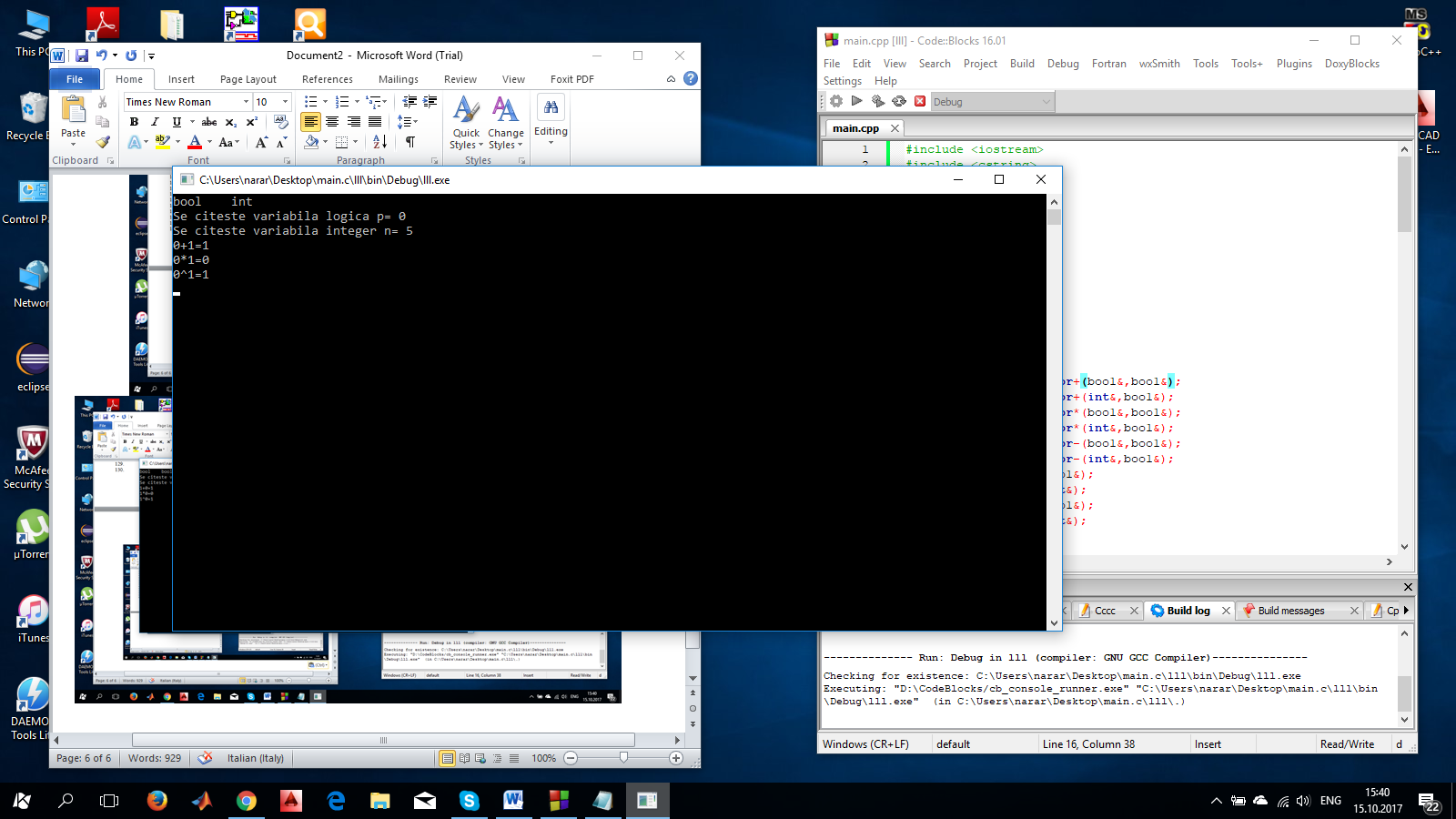
getch(); goto jump;

return 0;

}}







**b)**

#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 m;

int caut=0;

class Set{

public: int \*n;

int nr;

public:

Set()

{

n=new int[10];

for (int i=0;i<10;i++)

n[i]=0;

}

Set(int x)

{

nr=x;

n=new int[nr];

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

{

cout<<"Elementul "<<i<<" al setului: ";

cin>>n[i];

}

}

Set operator+(Set&);

Set operator\*(Set&);

Set operator-(Set&);

friend Set operator +=(Set&,int&);

friend int operator ==(Set&,Set&);

friend ostream &operator<<(ostream &,Set&);

friend istream &operator >>(istream&, Set&);

friend int apartenenta(Set&,int);

void show()

{

cout<<endl;

cout<<"Setul este : ";

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

cout<<n[i]<<" ";

}

};

istream &operator >> (istream &input, Set &f) {

int x;

cout << "nr. de elemente a setului: ";

cin>>x;

input.width(x);

for (int i = 0; i < x; i++) {

f.n[i] = input.get();

input.ignore(1);

}

return (input);

}

ostream &operator << (ostream &output, Set &p)

{

for (int i = 0; i <p.nr; i++) {

output << p.n[i] << ' ';

}

output << endl;

return (output);

}

Set Set::operator+(Set& x)

{

Set temp;

int w;

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

temp.n[i]=n[i];

w=10;

int t=nr;

for(int i=0;i<x.nr;i++)

{

w=0;

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

{

if(x.n[i]==temp.n[j]) w++;

}

if(w==0){temp.n[t]=x.n[i]; t++;

}

}

temp.nr=t;

return(temp);

}

Set Set::operator\*(Set& x)

{

int d,k=0;

d=nr+x.nr;

Set a;

for(int i=0;i<x.nr;i++)

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

if (x.n[i]==n[j])

{

a.n[k]=n[j]; k++;

}}

a.nr=k;

return(a);

}

Set Set:: operator-(Set& x)

{

int d,k=0,t,w;

d=nr+x.nr;

Set a,b;

for(int i=0;i<x.nr;i++)

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

if (x.n[i]==n[j])

{

a.n[k]=n[j]; k++;

}}

a.nr=k; t=0;

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

{ k=0;

for(int j=0;j<a.nr;j++){

if (n[i]==a.n[j]) k++;

}

if (k==0) {b.n[t]=n[i]; t++;}}

b.nr=t;

return(b);

}

int operator==(Set& x,Set& y)

{

int k;

if (x.nr=y.nr)

{

for(int i=0;i<x.nr;i++)

for(int j=0;j<y.nr;j++)

{

if(x.n[i]==y.n[j]) k++;

}

if(k>=x.nr) return 1;

else return 0;

}

else return 0;

}

Set operator+=(Set& x,int& m)

{

Set p;

int k;

cout<<"Pozitia:";

cin>>k;

int temp,temp1;

if(k==1)

{ temp=x.n[0];

x.n[0]=m;

for(int i=1;i<=x.nr;i=i+2)

{ m=x.n[i];

x.n[i]=temp;

temp=x.n[i+1];

x.n[i+1]=m;

}

x.nr=x.nr+1;

return (x);

}

else if(k==x.nr)

{

int t;

t=x.nr;

x.n[t]=m;

x.nr=t+1;

return(x);

}

else if(k>1)

{

for(int i=0;i<k-1;i++)

p.n[i]=x.n[i];

p.n[k-1]=m;

for(int i=k;i<=x.nr;i++)

p.n[i]=x.n[i-1];

p.nr=x.nr+1;

return(p);

}

}

void apartine(Set& x,int y)

{

for (int i=0;i<x.nr;i++)

{

if(y==x.n[i]) caut++;

}

}

int main(int argc, char \* const argv[])

{

int x,y;

cout<<"Nr. de elemente a setului 1:";

cin>>x;

cout<<"Nr. de elemente a setului 2:";

cin>>y;

Set a(x),b(y),c;

Set union1;

a.show();

b.show();

union1=a+b;

cout<<endl;

cout<<"Union:"<<union1;

Set inter;

inter=a\*b;

cout<<endl;

cout<<"Intresection:"<<inter;

Set dif;

dif=a-b;

cout<<endl;

cout<<"Diferenta a-b: "<<dif;

Set dif1;

dif1=b-a;

cout<<endl;

cout<<"Diferenta b-a: "<<dif1;

cout<<endl;

int z;

z=a==b;

if(z==0) cout<<"Seturile sunt egale:";

else cout<<"Seturile nu sunt egale:";

cout<<endl;

cout<<"Inserarea unui element";

int k,m;

cout<<endl;

cout<<"elementul inserat:";

cin>>m;

cout<<endl;

Set inser;

inser=b+=m;

cout<<inser;

cout<<"elementul cautat in setul 1: ";

cin>>m;

apartine(a,m);

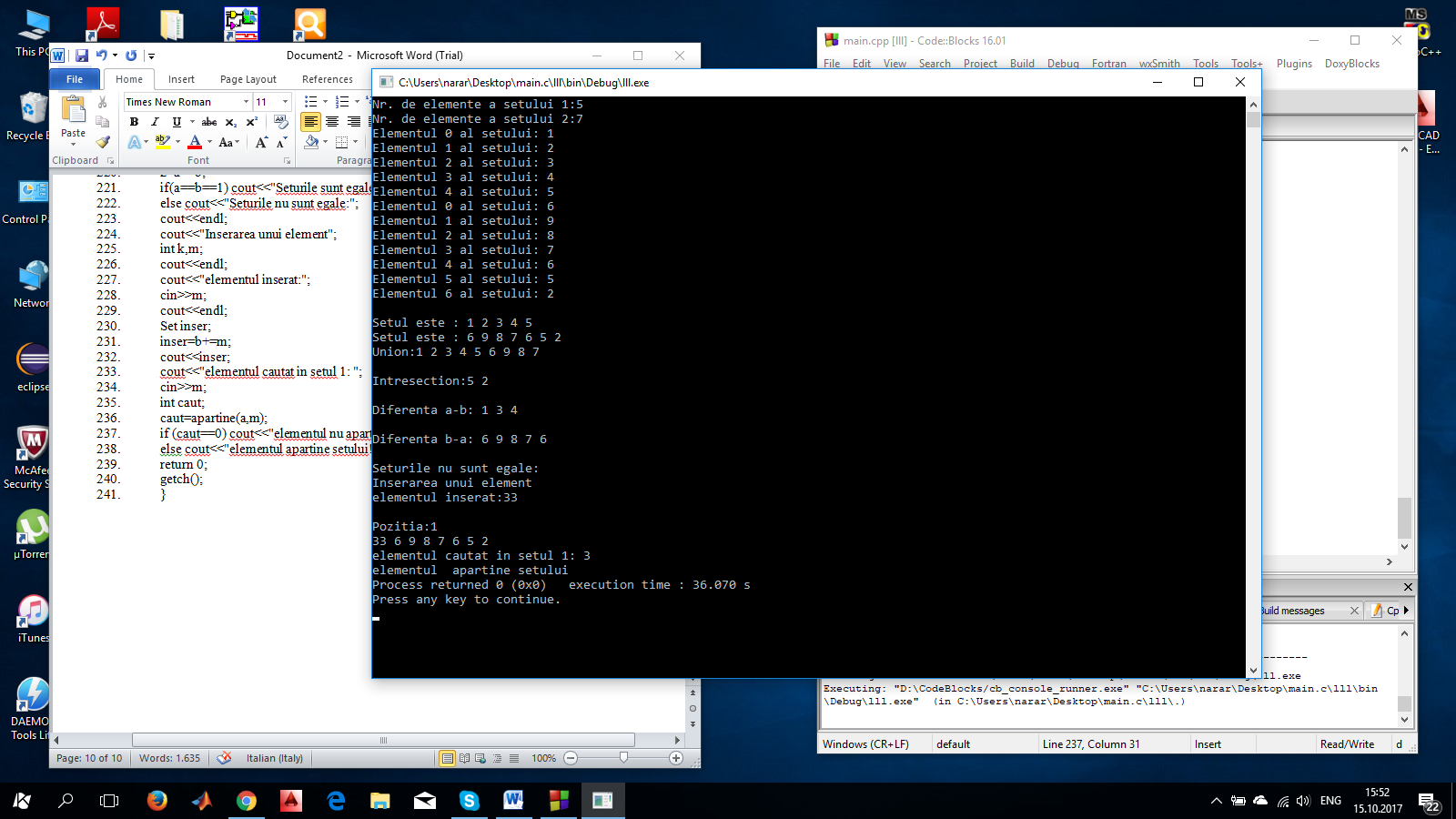
if(caut<1) cout<<"elementul nu apartine setului";

if(caut>=1) cout<<"elementul apartine setului!";

return 0;

getch();

}



**Concluzie:**

În urma efectuării lucrării de laborator nr.3 „ *Supraincarcarea operatorilor.*” , am însușit principiile supraîncărcării operatorilor în limbajul C++, această tehnică fiind foarte importantă deoarece permite simplificarea efectuării operațiilor tradiționale asupra obiectelor.

La fel am perfectionat tehnica de programare in limbajul C++ și anume lucrul asupra claselor, constructorilor și destructorilor.