Laborator 5

```
/* Lab 5  Popa Larisa-Ancuta  Prob 1

-să se scrie o aplicație C/C++ care folosește o structură de date cu numele Scerc care
conține raza ca
   și o variabilă de tip întreg
-intr-un program C/C++, declarați două variabile c1, c2 de tip Scerc
   și calculați aria și circumferința lor pentru valori ale razei introduse de la
tastatură cu două metode definite
-aceleași cerințe vor fi implementate în aceiasi aplicație folosind o clasă numită Cerc
cu atributul raza de tip private
  clasă ce va conține pe lângă metodele de calcul ale ariei și perimetrului un
constructor explicit cu parametru
  un destructor și o metodă de afișare raza
-extindeți aplicația astfel încât să definiți mai multe obiecte de tip Cerc la care să
accesați metodele specifice
   folosind obiectele instanțiate, pointeri la obiecte, referințe la obiecte
-introduceți o metodă de tip accesor, getRaza() care permite accesul la data privată raza
  si care sa o folositi pentru a afisa in main() raza obiectelor

*/

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

struct Scerc
{
	int raza;
};

class Cerc
{
private:
	int raza;

public:
	void set_raza(int);
	int get_raza();
	float aria();
	float circumferinta();
	void afisare_raza(int);
	Cerc(int r = 1); //constructor
	~Cerc(); //destructor
};

void Cerc::set_raza(int r)
{
	raza = r;
}

int Cerc::get_raza()
{
	return raza;
}
```

```
float Cerc::aria()
{
	return 3, 14 * (float)raza*(float)raza;
}

float Cerc::circumferinta()
{
	return 2 * 3, 14 * (float)raza;
}

void Cerc::afisare_raza(int raza)
{
	printf("\nRaza cercului este: %d", raza);
}

Cerc::Cerc(int r)
{
	raza = r;
}

Cerc::~Cerc()
{
	raza = 0;
}


float aria(struct Scerc);
float circumferinta(struct Scerc);

void main()
{
	Scerc c1, c2;

	cout << "\nIntroduceti raza cerc 1: ";
	cin >> c1.raza;
	cout << "\nIntroduceti raza cerc 2: ";
	cin >> c2.raza;

	cout << "\nArie cerc 1: " << aria(c1);
	cout << "\ncircumferinta cerc 1: " << circumferinta(c1);
	cout << "\nArie cerc 2: " << aria(c2);
	cout << "\ncircumferinta cerc 2: " << circumferinta(c2);

	Cerc c3, c4;
	int r3, r4;

	cout << "\n\nIntroduceti raza cerc 3: ";
	cin >> r3;
	c3.set_raza(r3);
	cout << "\nIntroduceti raza cerc 4: ";
	cin >> r4;
	c4.set_raza(r4);

	cout << "\nArie cerc 3: " << c3.aria();
	cout << "\ncircumferinta cerc 3: " << c3.circumferinta();
	cout << "\nArie cerc 4: " << c4.aria();
	cout << "\ncircumferinta cerc 4: " << c4.circumferinta() << endl;
```

```
}

float aria(struct Scerc c)
{
	return 3, 14 * (float)(c.raza)*(float)c.raza;
}

float circumferinta(struct Scerc c)
{
	return 2 * 3, 14 * (float)c.raza;
}
```

```
/* Lab 5  Popa Larisa-Ancuta  Prob 2

-să se definească o clasă numită myString (într-un fișier numit strClass.h)
 care să fie compusă din metodele specifice care efectuează următoarele operații pe
șiruri de caractere:

- determină lungimea șirului primit la intrare.
- determină ultima poziție de apariție a unui anumit caracter din șirul de intrare.
- returnează șirul primit la intrare, scris cu caractere majuscule.
- returnează șirul primit la intrare, scris cu caractere minuscule.
- returnează numărul de apariții ale unui anumit caracter din șirul primit.

*/

//main
#include "strClass.h"

int main()
{
	int n;
	char c1, c2;

	String();

	String sir;
	sir.citire();

	n = sir.lungime_sir();
	cout << "\nLungimea sirului este: " << n;

	cout << "\n\nIntroduceti un caracter: ";
	cin >> c1;
	cout << "Ultima pozitie de aparitie a caracterului '" << c1 << "' este: " <<
sir.ap_caracter(n, c1);

	cout << "\n\nSirul scris cu majuscule este : " << sir.majuscule(n);
	cout << "\n\nSirul scris cu minuscule este : " << sir.minuscule(n);

	cout << "\n\nIntroduceti un caracter: ";
	cin >> c2;
	cout << "Numarul de aparitii al caracterului '" << c2 << "' in sir este: " <<
sir.nr_ap(n, c2) << endl << endl;

	return 0;
```

```
}


//header
#pragma once

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string.h>

using namespace std;

class String
{
private:
	char *sir;
public:
	int lungime_sir(); //calculeaza lungimea sirului
	int ap_caracter(int, char); //afiseaza ultima pozitie a unui caracter
	char* majuscule(int); //returneaza situl cu majuscule
	char* minuscule(int); //returneaza sirul cu minuscule
	int nr_ap(int, char); //returneaza nr. de aparitii a unui caracter
	String(void); //constructor
	~String(); //destructor
	void citire();
};

int String::lungime_sir()
{
	return strlen(sir);
}

int String::ap_caracter(int n, char c)
{
	int i, poz = 0;

	for (i = 0;i<n;i++)
	{
		if (sir[i] == c)
			poz = i;
	}

	return poz;
}

char *String::majuscule(int n)
{
	int i;

	for (i = 0;i<n;i++)
		*(sir + i) -= 32;

	return sir;
}

char *String::minuscule(int n)
{
```

```
	int i;

	for (i = 0;i<n;i++)
		*(sir + i) += 32;

	return sir;
}

int String::nr_ap(int n, char c)
{
	int i, nr = 0;

	for (i = 0;i<n;i++)
	{
		if (*(sir + i) == c)
			nr++;
	}

	return nr;
}

String::String(void)
{
	sir = new char[256];
}

String::~String(void)
{
	delete[]sir;
}

void String::citire()
{
	cout << "Introduceti un sir de caractere: ";
	cin >> sir;
}
```

```
/* Lab 5  Popa Larisa-Ancuta  Prob 3

-să se scrie programul care citește de la tastatură un șir de maxim 10 caractere și care,
  pe baza clasei implementate anterior, efectuează asupra șirului de intrare operațiile
definite în cadrul clasei

*/

//main
#include "strClass.h"

#define DIM 10

int main()
{
	int n;
	char c1, c2;

	String();
```

```
	String sir;
	sir.citire();

	n = sir.lungime_sir();
	if (n <= 10)
	{
		cout << "\nLungimea sirului este: " << n;

		cout << "\n\nIntroduceti un caracter: ";
		cin >> c1;
		cout << "Ultima pozitie de aparitie a caracterului '" << c1 << "' este: "
<< sir.ap_caracter(n, c1);

		cout << "\n\nSirul scris cu majuscule este : " << sir.majuscule(n);
		cout << "\n\nSirul scris cu minuscule este : " << sir.minuscule(n);

		cout << "\n\nIntroduceti un caracter: ";
		cin >> c2;
		cout << "Numarul de aparitii al caracterului '" << c2 << "' in sir este: "
<< sir.nr_ap(n, c2) << endl << endl;
	}
	else
		cout << "\nDimensiunea sirului este prea mare!\n\n";

	return 0;
}

//header
#pragma once

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string.h>

using namespace std;

class String
{
private:
	char *sir;
public:
	int lungime_sir(); //calculeaza lungimea sirului
	int ap_caracter(int, char); //afiseaza ultima pozitie a unui caracter
	char* majuscule(int); //returneaza situl cu majuscule
	char* minuscule(int); //returneaza sirul cu minuscule
	int nr_ap(int, char); //returneaza nr. de aparitii a unui caracter
	String(void); //constructor
	~String(); //destructor
	void citire();
};

int String::lungime_sir()
{
	return strlen(sir);
}

int String::ap_caracter(int n, char c)
```

```
{
	int i, poz = 0;

	for (i = 0;i<n;i++)
	{
		if (sir[i] == c)
			poz = i;
	}

	return poz;
}

char *String::majuscule(int n)
{
	int i;

	for (i = 0;i<n;i++)
		*(sir + i) -= 32;

	return sir;
}

char *String::minuscule(int n)
{
	int i;

	for (i = 0;i<n;i++)
		*(sir + i) += 32;

	return sir;
}

int String::nr_ap(int n, char c)
{
	int i, nr = 0;

	for (i = 0;i<n;i++)
	{
		if (*(sir + i) == c)
			nr++;
	}

	return nr;
}

String::String(void)
{
	sir = new char[256];
}

String::~String(void)
{
	delete[]sir;
}

void String::citire()
{
	cout << "Introduceti un sir de caractere: ";
```

```
	cin >> sir;
}
```

```
/* Lab 5  Popa Larisa-Ancuta  Prob 4

-să se scrie programul care implementează clasa Numar cu un atribut de tip int val și
care, în cadrul funcției main(),
  declară un obiect de tipul clasei și apoi un pointer la acesta, prin intermediul cărora
se va afișa pe ecran rezultatul
  adunării a două numere de tip Numar cu valorile preluate de la tastatură in cadrul unor
obiecte Numar
-implementati metoda int suma_nr(Numar) care realizeaza suma în cadrul clasei și o
returneaza ca un int,
  metoda care insumeaza cele doua obiecte (curent si parametru)
-implementati metoda in cadrul clasei si alta metoda cu acelas scop, dar nume diferit, in
afara clasei

*/

//main
#include "Class.h"

int main()
{
	Numar nr1, nr2, si;
	int val, se;

	cout << "\nIntroduceti primul numar: ";
	cin >> val;
	nr1.set_val(val);

	cout << "\nIntroduceti al doilea numar: ";
	cin >> val;
	nr2.set_val(val);

	si = nr1.suma_nr(nr2);
	cout << "\nValoarea sumei calculate in interiorul clasei este : " << si.get_val();

	se = nr1.suma(nr1, nr2);
	cout << "\nValoarea sumei calculate in exteriorul clasei este: " << se << endl <<
endl;

	return 0;
}

//header
#pragma once

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

class Numar
{
private:
```

```
	int val;
public:
	Numar(int n = 1);
	~Numar(void);
	void set_val(int new_value)
	{
		val = new_value;
	}
	int get_val()
	{
		return val;
	}
	Numar suma_nr(Numar nr)
	{
		return val + nr.val;
	}
	int suma(Numar, Numar);
};

Numar::Numar(int n)
{
	val = n;
}

Numar::~Numar(void)
{
	val = 0;
}

int Numar::suma(Numar nr1, Numar nr2)
{
	return nr1.val + nr2.val;
}
```

```
/* Lab5  Popa Larisa-Ancuta  Prob 5

-să se definească o clasă care implementează metodele:

- int plus(int x, int y), care returnează suma valorilor primite la apelul metodei;
- int minus(int x, int y), care returnează diferenta valorilor primite la apelul metodei;
- int inmultit(int x, int y), care returnează produsul valorilor primite la apelul
metodei;
- float impartit(int x, int y), care returnează catul valorilor primite la apelul
metodei;

-apoi să se scrie aplicaţia care utilizează această clasă
-considerati si cazul in care in cadrul clasei aveti atributele de tip int x si y, caz in
care
  metodele nu vor mai avea parametrii

-observaţie: În cazul împărţirii, trebuie verificată validitatea operaţiei (împărţitor
diferit de zero)
-in cazul în care operaţia este imposibilă, trebuie afişat un mesaj de eroare

*/
```

```cpp
//main
#include "Class.h"

int main()
{
	Operatii op;
	int x, y;

	cout << "\nIntroduceti doua numere intregi:\n";
	cin >> x >> y;
	op.set_xy(x, y);

	cout << "\nAdunare: " << op.plus(x, y);
	cout << "\nScadere: " << op.minus(x, y);
	cout << "\nInmultire: " << op.inmultire(x, y);

	if (y != 0)
		cout << "\nImpartire: " << op.impartire(x, y) << endl << endl;
	else
		cout << "\nOperatia de impartire nu se poate efectua!\n\n";
	return 0;
}
```

```cpp
//header
#pragma once

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

class Operatii
{
	int x, y;
public:
	Operatii(int x1 = 0, int y1 = 1)
	{
		x = x1;
		y = y1;
	}
	void set_xy(int x1,int y1)
	{
		x = x1;
		y = y1;
	}
	int get_x()
	{
		return x;
	}
	int get_y()
	{
		return y;
	}

	int plus(int x, int y);
	int minus(int x, int y);
	int inmultire(int x, int y);
	float impartire(int x, int y);
```

```
};

int Operatii::plus(int x, int y)
{
	return x + y;
}
int Operatii::minus(int x, int y)
{
	return x - y;
}
int Operatii::inmultire(int x, int y)
{
	return x * y;
}
float Operatii::impartire(int x, int y)
{
	return (float)x / y;
}
```

```
/*  Lab 5  Popa Larisa-Ancuta  Prob 6

-să se creeze o clasă care să modeleze numerele complexe
-scrieţi un program care utilizează această clasă si defineste doua obiecte
  afisand caracteristicile obiectelor si rezultatele operatiilor definite
-Folositi exemplul 3 cu rezultat in obiectul curent

*/

//main
#include "Class.h"

int main()
{
	Complex o1, o2;
	double val;

	cout << "\nPartea reala obiect 1:";
	cin >> val;
	o1.set_Re(val);
	cout << "\nPartea imaginara obiect 1:";
	cin >> val;
	o1.set_Im(val);

	cout << "\nPartea reala obiect 2:";
	cin >> val;
	o2.set_Re(val);
	cout << "\nPartea imaginara obiect 2:";
	cin >> val;
	o2.set_Im(val);

	cout << "\nPartea reala a obiectului 1 este: " << o1.get_Re();
	cout << "\nPartea imaginara a obiectului 1 este:" << o1.get_Im();
	cout << "\nModulul obiectului 1 este: " << o1.modul();
	cout << "\nFaza obiectului 1 este: " << o1.faza();

	cout << "\n\nPartea reala a obiectului 2 este: " << o2.get_Re();
```

```
	cout << "\nPartea imaginara a obiectului 2 este: " << o2.get_Im();
	cout << "\nModulul obiectului 2 este: " << o2.modul();
	cout << "\nFaza obiectului 2 este: " << o2.faza() << endl << endl;
	return 0;
}
```

```
//header
#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <math.h>

using namespace std;

class Complex
{
	double re, im;
public:
	Complex(double x = 0.0, double y = 0.0)
	{
		re = x;
		im = y;
	}
	void set_Re(double real)
	{
		re = real;
	}
	double get_Re()
	{
		return re;
	}
	void set_Im(double imaginar)
	{
		im = imaginar;
	}
	double get_Im() {
		return im;
	}
	double modul()
	{
		return sqrt(re*re + im*im);
	}
	double faza()
	{
		return atan2((re), (im));
	}
};
```

```
/* Lab 5  Popa Larisa-Ancuta  Prob 7

-să se scrie un program care implementează clasa Aritmetica cu două atribute a și b de
tip numeric (int, float sau double)
  si metode setter si getter adecvate
-implementați metoda suma() în interiorul clasei și metoda diferenta() ce apartine de
asemenea clasei,
```

```
  dar e definita în afara clasei, metode care vor fi apelate prin intermediul unui obiect
al clasei Aritmetica
-in funcția principală main() instanțiați trei obiecte de tip Aritmetica
-modificati atributele a si b la fiecare obiect in parte folsind metodele de tip setter
-aplicați asupra lor operațiile de adunare și scădere pe care le-ați implementat prin
metodele suma() și diferenta()
-metodele returneaza valorile numerice corespunzatoare operatiei folosind cele doua
atribute ale clasei valori
  ce le veti afisa in main()
-la fiecare grup de operatii adunare/scadere afisati valorile atributelor obiectului
folosind metodele de tip getter

*/

//main
#include "Class.h"

int main()
{
	Aritmetica o1, o2, o3;
	int a, b;

	cout << "\nObiectul 1\n" << " a=";
	cin >> a;
	cout << " b=";
	cin >> b;
	o1.set_nr(a, b);
	cout << "Suma: " << o1.suma();
	cout << "\nDiferenta: " << o1.diferenta();

	cout << "\n\nObiectul 2 \n" << " a=";
	cin >> a;
	cout << " b=";
	cin >> b;
	o2.set_nr(a, b);
	cout << "Suma: " << o2.suma();
	cout << "\nDiferenta: " << o2.diferenta();

	cout << "\n\nObiectul 3 \n" << " a=";
	cin >> a;
	cout << " b=";
	cin >> b;
	o3.set_nr(a, b);
	cout << "Suma: " << o3.suma();
	cout << "\nDiferenta: " << o3.diferenta() << endl << endl;

	return 0;
}

//header
#pragma once

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

class Aritmetica
```

```
{
private:
	int a, b;
public:
	Aritmetica(int a1 = 0, int b1 = 0);
	void set_nr(int a1, int b1);
	int suma();
	int diferenta();
};

Aritmetica::Aritmetica(int a1, int b1)
{
	a = a1;
	b = b1;
}

void Aritmetica::set_nr(int a1, int b1)
{
	a = a1;
	b = b1;
}

int Aritmetica::suma()
{
	return a + b;
}

int Aritmetica::diferenta()
{
	return a - b;
}
```

```
/* Lab 5  Popa Larisa-Ancuta  Prob 8

-pornind de la clasa Complex, ex.4, să se implementeze operațiile de adunare, scădere,
înmulțire și împărțire
  pentru numere complexe prin metode corespunzătoare implementate la alegere în clasă
și/sau în afara ei
-testați aceste metode prin instanțierea unor obiecte
-metodele vor returna obiecte de tip Complex si in main() vor fi afisate rezultatele
folosind metode accesor

*/

//main
#include "Class.h"

int main()
{
	Complex z1, z2;
	double re, im;

	cout << "\nObiectul 1: \n" << "  Re= ";
	cin >> re;
	z1.set_Re(re);
	cout << "  Im= ";
```

```
	cin >> im;
	z1.set_Im(im);

	cout << "\nObiectul 2: \n" << "  Re= ";
	cin >> re;
	z2.set_Re(re);
	cout << "  Im= ";
	cin >> im;
	z2.set_Im(im);

	Complex rez=z1.suma(z2);
	cout << "\n\nAdunare\n   Re=" << rez.get_Re() << "\n   Im=" << rez.get_Im();

	rez=z1.diferenta(z2);
	cout << "\n\nScadere\n   Re=" << rez.get_Re() << "\n   Im=" << rez.get_Im();

	rez=z1.inmultire(z2);
	cout << "\n\nInmultire\n   Re=" << rez.get_Re() << "\n   Im=" << rez.get_Im() <<
endl << endl;


	return 0;
}
```

```
//header
#pragma once

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

class Complex
{
	float re, im;

public:
	Complex(float x = 0.0, float y = 0.0)
	{
		re = x;
		im = y;
	}
	void set_Re(float x)
	{
		re = x;
	}
	void set_Im(float x)
	{
		im = x;
	}
	float get_Re()
	{
		return re;
	}
	float get_Im()
	{
		return im;
	}
```

```
        Complex suma(Complex z);
        Complex diferenta(Complex z);
        Complex inmultire(Complex z);
};

Complex Complex::suma(Complex z)
{
        Complex sum;

        sum.im = z.im + im;
        sum.re = z.re + re;

        return sum;
}

Complex Complex::diferenta(Complex z)
{
        Complex dif;

        dif.im = im - z.im;
        dif.re = re - z.re;

        return dif;
}

Complex Complex::inmultire(Complex z)
{
        Complex prod;

        prod.im = z.im*re + z.re*im;
        prod.re = z.re*re - z.im*im;

        return prod;
}
```

```
/*  Lab 5  Popa Larisa-Ancuta  Prob 9

-declarati o clasa Fractie care are doua atribute intregi de tip private a si b pentru
numarator si numitor
-definiti doua metode de tip set() respectiv get() pentru fiecare din atributele clasei
-instantiati doua obiecte de tip Fractie si afisati atributele initiale si cele obtinute
dupa folosirea metodelor set()
-definiti o metoda simplifica() apelata cu un obiect pentru care au fost apelate metodele
de tip set(),
  care determina divizorii numitorului si numaratorului, ii afiseaza si realizeaza
simplificarea fractiei,
  afisand in metoda si rezultatul obtinut (noua fractie numarator/numitor)

*/

//main
#include "Class.h"

int main()
{
        Fractie f;
```

```
	int a, b;

	cout << "\nIntroduceti valoarea numitorului: ";
	cin >> a;
	f.set_a(a);
	cout << "\nIntroduceti valoarea numaratorului: ";
	cin >> b;
	f.set_b(b);

	cout << "\n\nFractia inainte de simplificare: " << f.get_a() << "/" << f.get_b();

	f.simplificare(f);
	cout << "\n\nFractia dupa simplificare: " << f.get_a() << "/" << f.get_b() << endl
<< endl;

	return 0;
}

//main
#pragma once

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

class Fractie {
private:
	int a, b;
public:
	Fractie(int x = 0, int y = 1)
	{
		a = x;
		b = y;
	};

	void set_a(int new_value)
	{
		a = new_value;
	};
	void set_b(int new_value)
	{
		b = new_value;
	};

	int get_a()
	{
		return a;
	};
	int get_b()
	{
		return b;
	};
	void simplificare(Fractie &);
};

int cmmdc(int x, int y)
{
```

```
	if (!y)
		return x;
	return cmmdc(y, x%y);
}

void Fractie::simplificare(Fractie &f)
{
	int d = cmmdc(f.a, f.b);
	f.a /= d;
	f.b /= d;
}
```