Laborator 6

```
/* Lab 6  Popa Larisa-Ancuta  Prob 1

-să se scrie o aplicație C++ care implementează o clasă numită PilotF1
-clasa definește variabilele private nume (șir de caractere), echipa (șir de caractere),
varsta (int),
  record (int), nr_pole_position (int)
-ca membri publici, clasa conține metode accesor/getter și mutator/setter distincte
pentru fiecare din atributele clasei

-in funcția main( ), să se creeze 3 instanțe distincte ale clasei PilotF1
-să se folosească metodele mutator/setter pentru a inițializa datele din fiecare obiect
cu informația corespunzătoare
 citită de la tastatură
-folosind metodele accesor/getter, să se afișeze toate datele pilotului cu cel mai bun
record

*/

//main
#define  _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

#include  "Class.h"

PilotF1 citire(); //citire si setare date pilot
void afisare(PilotF1); //afisare date pilot
void best_pilot(PilotF1, PilotF1, PilotF1); //afisare pilot cu cel mai bun record

int main()
{
	PilotF1  p1, p2, p3;

	cout << "Introduceti datele primului pilot:";
	p1 = citire();

	cout << "\nIntroduceti datele celui de al doilea pilot:";
	p2 = citire();

	cout << "\nIntroduceti datele celui de al treilea pilot:";
	p3 = citire();

	cout << "\n\nPilotul 1:\n";
	afisare(p1);

	cout << "\nPilotul 2:\n";
	afisare(p2);

	cout << "\nPilotul 3: \n";
	afisare(p3);

	best_pilot(p1, p2, p3);

	return 0;
}
```

```
PilotF1 citire()
{
	PilotF1 p;
	char n[DIM], e[DIM];
	int v, r, nr;

	cout << "\nNume: ";
	cin >> n;
	p.set_nume(n);

	cout << "Echipa: ";
	cin >> e;
	p.set_echipa(e);

	cout << "Varsta: ";
	cin >> v;
	p.set_varsta(v);

	cout << "Record: ";
	cin >> r;
	p.set_record(r);

	cout << "Nr. pole position: ";
	cin >> nr;
	p.set_nr_pp(nr);

	return p;
}

void afisare(PilotF1 p)
{
	cout << "  Nume: " << p.get_nume();
	cout << "\n  Echipa:" << p.get_echipa();
	cout << "\n  Varsta: " << p.get_varsta();
	cout << "\n  Record: " << p.get_record();
	cout << "\n  Nr.Pole  position: " << p.get_nr_pp() << endl << endl;
}

void best_pilot(PilotF1 p1, PilotF1 p2, PilotF1 p3)
{
	int r1 = p1.get_record(), r2 = p2.get_record(), r3 = p3.get_record();

	cout << "\nPilotul cu cel mai bun record este:\n";
	if (r1 < r2 && r1 < r3)
		afisare(p1);
	else
	{
		if (r2 < r1 && r2 < r3)
			afisare(p2);
		else
		{
			if (r3 < r1 && r3 < r2)
				afisare(p3);
		}
	}
}
```

```
//header Class.h
#pragma once

#define DIM 30

class  PilotF1
{

private:
	char  nume[DIM], echipa[DIM];
	int  varsta, record, nr_pole_position;

public:
	PilotF1()
	{
		strcpy(nume, "");
		strcpy(echipa, "");
		varsta = 0;
		record = 0;
		nr_pole_position = 0;
	}

	void set_nume(char*);
	void set_echipa(char*);
	void set_varsta(int);
	void set_record(int);
	void set_nr_pp(int);

	char*  get_nume();
	char*  get_echipa();
	int  get_varsta();
	int  get_record();
	int  get_nr_pp();
};

void PilotF1::set_nume(char *n)
{
	if (n != 0)
		strncpy(nume, n, DIM);
	else
		strcpy(nume, "-");
}

void PilotF1::set_echipa(char *e)
{
	if (e != 0)
		strncpy(echipa, e, DIM);
	else
		strcpy(echipa, "-");
}

void PilotF1::set_varsta(int v)
{
	varsta = v;
}

void PilotF1::set_record(int r)
{
```

```
	record = r;
}

void PilotF1::set_nr_pp(int nr)
{
	nr_pole_position = nr;
}

char*  PilotF1::get_nume(void)
{
	return nume;
}

char*  PilotF1::get_echipa(void)
{
	return echipa;
}

int  PilotF1::get_varsta(void)
{
	return varsta;
}

int  PilotF1::get_record(void)
{
	return record;
}

int  PilotF1::get_nr_pp(void)
{
	return  nr_pole_position;
}
```

```
/*  Lab 6  Popa Larisa-Ancuta  Prob 2

-să se modifice exemplul 2 astfel încât codul să poată fi lansat în execuție considerand
atributul clasei private

*/

#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

class Test2
{
	int x;
public:
	Test2()
	{
		cout << "\nApel constructor explicit vid.";
	}
	void set_val(int nr)
	{
		x = nr;
```

```
	}
	int get_val()
	{
		return x;
	}
};

int main()
{
	Test2 ob1;
	int a;
	cout << "\nIntroduceti valoarea variabilei de tip \"int\" din clasa: ";
	cin >> a;
	ob1.set_val(a);

	cout << "\nValoarea introdusa este: " << ob1.get_val() << endl << endl;

	return 0;
}
```

```
/*  Lab 6  Popa Larisa-Ancuta  Prob 3

-ornind de la exemplul care tratează lucrul cu matrice in varianta transformata cu
alocare dinamica
-completați codul scris cu metodele specifice pentru:
    - afișarea elementelor de pe diagonala secundara a matricei, dacă matricea este
pătratică;
          în caz contrar se afișează un mesaj corespunzător;
    - afișarea elementelor de sub diagonala principala;
    - afișarea unei matrice de dimensiunea celei inițiale ale cărei elemente pot avea
valori de 0
          (dacă elementul corespunzător este mai mare decât o valoare citita) sau 1 (în
caz contrar)

*/

//main
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

#include "Class.h"

int main()
{
	int dim1, dim2;

	cout << "\nIntroduceti dimensiunile 1 si 2 ale matricii: (<=10):\n";
	cin >> dim1;
	cin >> dim2;

	Matrix m1(dim1, dim2); //instantiere cu citire valori

	m1.displayMatrix();
```

```
	int c;
	cout << "\nIntroduceti un numar de coloana ale carei elemente vor fi afisate:
(<dim2) ";
	cin >> c;
	m1.displayColumn(c);
	//incercare (imposibila) de a accesa direct un membru privat al clasei
	//m1.returnElement(0, 0);

	cout << endl << endl;

	m1.afisare_DiagS();

	m1.afisare_subDiagP();

	int val;
	cout << "Introduceti o valoare intreaga: ";
	cin >> val;
	m1.afisare_newMat(val);

	return 0;
}

//header Class.h
#pragma once

const int Max1 = 10;
const int Max2 = 10;

class Matrix
{
	//atribute
	int matrix[Max1][Max2], dim1, dim2;

	//declararea metodei de afisare a unui element
	int returnElement(int row, int column);
public:
	//constructor explicit cu parametri – recomandat a folosi o metoda diferita a citi
valorile
	Matrix(int dim1, int dim2)
	{
		//variabile locale
		int i, j;
		this->dim1 = dim1;
		this->dim2 = dim2;
		cout << "\nIntroduceti elementele matricii: ";
		for (i = 0; i<dim1; i++)
		{
			for (j = 0; j<dim2; j++)
			{
				cout << "\nmatrix[" << i << "][" << j << "] = ";
				cin >> matrix[i][j];
			}
		}
	}

	//metoda de afisare a matricii din clasa; implementare in cadrul clasei
	void displayMatrix()
	{
```

```
		//variabile locale
		int i, j;
		cout << "\nElementele matricii: ";
		for (i = 0; i<dim1; i++)
		{
			cout << "\n";
			for (j = 0; j<dim2; j++)
			{
				//apelul metodei private care returneaza valoarea unui element
din matrice
				cout << returnElement(i, j) << " ";
			}
		}
		cout << endl;
	}

	//declararea metodei de afisare a elementelor unei coloane
	void displayColumn(int col);

	void afisare_DiagS();
	void afisare_subDiagP();
	void afisare_newMat(int);
};

//implementarea externa a metodelor (publice sau private) declarate in clasa
void Matrix::displayColumn(int col) {
	if (col<0 || col >= dim2) {
		cout << "\nColoana cu numarul " << col << " nu exista in matricea din
clasa!\n";
	}
	else {
		cout << "\nElementele coloanei " << col << ": ";
		for (int i = 0; i<dim1; i++) {
			cout << returnElement(i, col) << " ";
		}
	}
}

int Matrix::returnElement(int row, int column) {
	return matrix[row][column];
}

void Matrix::afisare_DiagS()
{
	int i;

	if (dim1 == dim2)
	{
		cout << "Elementele de pe diagonala secundara sunt: \n";

		for (i = 0;i < dim1;i++)
			cout << returnElement(i, dim1 - i - 1) << "  ";
	}
	else
		cout << "\nMatricea nu este patratica!\n\n";

	cout << endl << endl;
}
```

```
void Matrix::afisare_subDiagP()
{
	int i, j;

	if (dim1 == dim2)
	{
		cout << "Elementele de sub diagonala principala sunt: \n";

		for (i = 0;i < dim1;i++)
		{
			for (j = 0;j < dim2;j++)
			{
				if (i > j)
					cout << returnElement(i, j) << "  ";
			}
		}
	}
	else
		cout << "\nMatricea nu este patratica!\n\n";

	cout << endl << endl;
}

void Matrix::afisare_newMat(int val)
{
	int i, j;

	cout << "Matricea noua: \n";
	for (i = 0;i < dim1;i++)
	{
		for (j = 0;j < dim2;j++)
		{
			if (val > returnElement(i, j))
				cout << "0 ";
			else
				cout << "1 ";
		}
		cout << endl;
	}

	cout << endl << endl;
}
```

```
/* Lab 6  Popa Larisa-Ancuta  Prob 4

-să se scrie o clasă care are ca variabilă privată un câmp de tip dată, definit într-o
structură externă clasei
    (zi – int, luna – int, an - int)
-clasa conține metode mutator/setter și accesor/getter (publice) pentru informația
privată
-in clasă se mai află doua metode publice care:
   - testează validitatea datei stocate;
   - scrie într-un fișier toate datele din anul curent care preced (cronologic) data
stocată în clasă;
```

```
-in functia main( ), după instanțierea clasei și citirea de la tastatură a componentelor
unei date
-să se apeleze metodele membre și apoi să se verifice rezultatele obținute

*/
//main
#include <iostream>
#include <fstream>

using namespace std;

extern struct Dt
{
	int zi, luna, an;
};

ofstream fout("File.txt");

#include "Class.h"

int main()
{
	Data data;
	int z, l, a;

	cout << "Introduceti o data calendaristica: \n";
	cout << "  Zi: ";
	cin >> z;
	cout << "  Luna: ";
	cin >> l;
	cout << "  An: ";
	cin >> a;

	data.set_data(z, l, a);

	Dt data2 = data.get_data();

	if (data.validare_data(data))
	{
		cout << "\nData introdusa este: " << data2.zi << "/" << data2.luna << "/"
<< data2.an << endl << endl;

		data.afisare(data);
	}
	else
		cout << "\nData incorecta!\n\n";

	fout.close();
	return 0;
}

//header Class.h
#pragma once

class Data
{
private:
	Dt d;
```

```
public:
	Data()
	{
		d.zi = 0;
		d.luna = 0;
		d.an = 0;
	};

	void set_data(int, int, int);
	Dt get_data();
	int validare_data(Data);
	void afisare(Data);
};

void Data::set_data(int z, int l, int a)
{
	d.zi = z;
	d.luna = l;
	d.an = a;
}

Dt Data::get_data()
{
	return d;
}

int Data::validare_data(Data data)
{
	if (data.d.an > 0 && (data.d.luna > 0 && data.d.luna <= 12) && (data.d.zi > 0 &&
data.d.zi <= 31))
		return 1;

	return 0;
}

void Data::afisare(Data data)
{
	int i, j, n;

	for (i = 1;i < data.d.luna;i++)
	{
		if (i == 2)
			n = 28;
		else
		{
			if (i % 2 == 0)
				n = 30;
			else
				n = 31;
		}

		for (j = 1;j <= n;j++)
			fout << "\n" << j << "/" << i << "/" << data.d.an;
	}

	for (i = 1; i < data.d.zi; i++)
		fout << "\n" << i << "/" << data.d.luna << "/" << data.d.an;
}
```

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

-să se scrie o aplicație C++ care implementează o clasă numită Triunghi
-clasa cuprinde atributele private pentru laturile a, b, c, un constructor cu parametrii,
metode setter si getter adecvate
-calculați aria și perimetrul prin metode specifice clasei
-scrieți o metodă care să indice dacă triunghiul este dreptunghic sau nu
-definiti o metoda private cu parametrii in clasa care permite verificarea condiției ca
laturile să formeze un triunghi
-ea va fi folosita si de metodele setter

*/

//main
#include <iostream>

using namespace std;

#include "Class.h"

int main()
{
	Triunghi tr;
	int a, b, c;

	cout<<"Introduceti valorile laturilor triunghiului:\n ";
	cout << " a=";
	cin >> a;
	cout << "  b=";
	cin >> b;
	cout << "  c=";
	cin >> c;

	tr.set_laturi(a, b, c);

	if (tr.verif_triunghi())
	{
		cout << "\nValorile introduse pot forma un triunghi.\n";

		cout << "\nAria triunghiului este: " << tr.aria() << endl;
		cout << "\nPerimetrul triunghiului este: " << tr.perimetru() << endl;

		if (tr.verif_dreptunghic())
			cout << "\nTriunghiul este dreptunghic!\n\n";
		else
			cout << "\nTriunghiul nu este dreptunghic!\n\n";
	}
	else
		cout << "\nValorile introduse nu pot forma un triunghi!\n\n";

	return 0;
}

//header Class.h
#pragma once

class Triunghi
{
```

```
private :
	int a, b, c;
public:
	Triunghi()
	{
		a = 0;
		b = 0;
		c = 0;
	}

	void set_laturi(int, int, int);
	int get_a();
	int get_b();
	int get_c();

	int perimetru();
	float aria();
	int verif_dreptunghic();
	int verif_triunghi();

};

void Triunghi::set_laturi(int l1, int l2, int l3)
{
	a = l1;
	b = l2;
	c = l3;
}

int Triunghi::get_a()
{
	return a;
}

int Triunghi::get_b()
{
	return b;
}

int Triunghi::get_c()
{
	return c;
}

int Triunghi::perimetru()
{
	return a + b + c;
}

float Triunghi::aria()
{
	float p = (float)perimetru() / 2;

	return sqrt(p*(p - a)*(p - b)*(p - c));
}

int Triunghi::verif_dreptunghic()
{
```

```
	int a2 = pow(a, 2), b2 = pow(b, 2), c2 = pow(c, 2);

	if (a2 == b2 + c2 || b2 == a2 + c2 || c2 == a2 + b2)
		return 1;

	return 0;
}

int Triunghi::verif_triunghi()
{
	if (a > 0 && b > 0 && c > 0 && (a + b > c) && (b < a + c) && (b + c > a))
		return 1;

	return 0;
}
```

```
/* Lab 6  Popa Larisa-Ancuta  Prob 7

-să se scrie clasa Seif, cu atributele private cifru și suma
-descrieți metodele private getSuma() și setSuma( ) și metodele publice puneInSeif( ) și
scoateDinSeif( ) cu care
   să accesați suma de bani care se află în seif
-metoda puneInSeif( ) poate apela getSuma() și setSuma( ), metoda scoateDinSeif( ) poate
apela getSuma( ) și setSuma( )
-instanțiați obiecte din clasa Seif, iar metodele puneInSeif( ) și scoateDinSeif( ) vor
putea accesa suma doar dacă
   parametrul de tip cifru utilizat corespunde obiectului instanțiat
-in caz de diferență de cifru, se va da un mesaj

*/

//main
#include <iostream>

using namespace std;

#include "Class.h"

int main()
{
	Seif s(9876, 0);

	s.puneInSeif();

	s.scoateDinSeif();

	return 0;
}

//header Class.h
#pragma once

class Seif
{
private:
	int cifru;
```

```
        float suma;
        void setSuma(float);
        float getSuma();
public:
        Seif()
        {
                cifru = 0;
                suma = 0;
        }
        Seif(int c, float s)
        {
                cifru = c;
                suma = s;
        }
        void puneInSeif();
        void scoateDinSeif();
};

void Seif::setSuma(float val)
{
        suma = val;
}

float Seif::getSuma()
{
        return suma;
}

void Seif::puneInSeif()
{
        int pin;
        float sum;

        cout << "\nIntroduceti cifrul seifului: \n";
        cin >> pin;

        if (pin == cifru)
        {
                cout << "\nIntroduceti suma pe care doriti sa o depuneti: \n";
                cin >> sum;

                setSuma(getSuma() + sum);

                cout << "\nSlod cont: " << getSuma() << endl << endl;
        }
        else
                cout << "\n\nCifru incorect!\n\n";
}

void Seif::scoateDinSeif()
{
        int pin;
        float sum;

        cout << "\nIntroduceti cifrul seifului: \n";
        cin >> pin;

        if (pin == cifru)
```

```
	{
		cout << "\nIntroduceti suma pe care doriti sa o retrageti: \n";
		cin >> sum;

		setSuma(getSuma() - sum);

		cout << "\nSlod cont: " << getSuma() << endl << endl;
	}
	else
		cout << "\n\nCifru incorect!\n\n";
}
```

```
/*  Lab 6  Popa Larisa-Ancuta  Prob 8

Dezvoltați aplicația prezentată în exemplul 6 prin:
- utilizarea valorilor returnate de metoda setValidCnp( ) pentru a valida suplimentar
(luna si ziua) CNP-ul in main( )
- permiterea intoducerii de coduri CNP care încep cu alte cifre decât 1 și 2, cu
analizarea semnificației noilor valori
   (5 – masculin, 6 - feminin)

*/
//main
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <time.h>

using namespace std;
#include "Class.h"

int main()
{
	Person p1;
	char aux_string[30];

	cout << "\nEnter Name: ";
	cin >> aux_string;//Popescu
	p1.set_Nume(aux_string);

	cout << "\nEnter SurName: ";
	cin >> aux_string;//Bitanescu
	p1.set_Prenume(aux_string);

	cout << "\nEnter CNP: ";
	cin >> aux_string;//1890403120671
	p1.setValidCNP(aux_string);

	cout << "\nDate despre obiect: " << endl;
	cout << "\tNume: " << p1.get_nume() << "\n\tPrenume: " << p1.get_prenume()
<<"\n\tCNP: " << p1.get_CNP() << endl;
	cout << "\tSex: " << p1.get_gen() << endl;
	cout << "\tData nasterii: " <<
		p1.get_an_nast() << "/" << p1.get_luna_nast() << "/" << p1.get_zi_nast() <<
endl;
	cout << "\tVarsta: " << p1.get_varsta() << endl;
}
```

```
//header Class.h
#pragma once

const int dim = 24;

class Person
{
	char nume[dim];
	char prenume[dim];
	char CNP[14];

public:
	void set_Nume(char*);
	void set_Prenume(char*);
	int setValidCNP(char*);

	char* get_nume(void);
	char* get_prenume(void);
	char* get_CNP(void);

	char get_gen(void);
	int get_an_nast(void);
	int get_luna_nast(void);
	int get_zi_nast(void);
	int get_varsta(void);
};


void Person::set_Nume(char *n)
{
	if (n != 0)
		strncpy(nume, n, 15);
	else
		strcpy(nume, "Necunoscut");
}

void Person::set_Prenume(char *p)
{
	if (p != 0)
		strncpy(prenume, p, 23);
	else
		strcpy(prenume, "Necunoscut");
}

int Person::setValidCNP(char *c)
{
	char buf[3];
	int n;
	if (c != 0)
	{
		// validare CNP: numai pentru cifra gen, cifrele pentru AN, LU, ZI
		if (strlen(c) != 13)//lungime cnp
			return 1;
		if (c[0] != '1' && c[0] != '2'&& c[0] != '5'&& c[0] != '6')//cod cnp
			return 2;

		strncpy_s(buf, c + 1, 2);//an
		buf[2] = '\0';
```

```
                n = atoi(buf);
                if (n > 99)
                        return 3;//inconsistent

                strncpy_s(buf, c + 3, 2);//luna
                buf[2] = '\0';
                n = atoi(buf);
                if (n == 0 || n > 12)
                        return 4;

                strncpy_s(buf, c + 5, 2);//zi
                buf[2] = '\0';
                n = atoi(buf);
                if (n == 0 || n > 31)
                        return 5;

                strcpy_s(CNP, c);//copiere sir c valid in cnp

                return 0;
        }
        else return -1;
}

char* Person::get_nume()
{
        return nume;
}

char* Person::get_prenume()
{
        return prenume;
}

char* Person::get_CNP()
{
        return CNP;
}

char Person::get_gen(void)
{
        if (CNP[0] == '1' || CNP[0]=='5') return 'M';
        if (CNP[0] == '2'|| CNP[0] == '6') return 'F';

        return 'X';
}

int Person::get_an_nast(void)
{
        char buf[3];
        strncpy(buf, CNP + 1, 2);
        buf[2] = '\0';

        if(CNP[0] == '1' || CNP[0] == '2')
                return(1900 + atoi(buf));
        else
                if(CNP[0] == '5' || CNP[0] == '6')
                        return(2000 + atoi(buf));
```

```
	return -1;
}

int Person::get_luna_nast(void)
{
	char buf[3];
	strncpy(buf, CNP + 3, 2);
	buf[2] = '\0';
	return(atoi(buf));
}

int Person::get_zi_nast(void)
{
	return((CNP[5] - '0') * 10 + (CNP[6] - '0'));
}

int Person::get_varsta(void)
{
	struct tm *newTime;
	time_t szClock;
	time(&szClock);
	newTime = localtime(&szClock);
	int an_c = 1900 + newTime->tm_year;
	int an_n = get_an_nast();
	int n = an_c - an_n;
	int lu_c = newTime->tm_mon + 1;
	int lu_n = get_luna_nast();
	if (lu_c < lu_n) n--;
	else {
		if (lu_c == lu_n) {
			int zi_c = newTime->tm_mday;
			int zi_n = get_zi_nast();
			if (zi_c < zi_n)
				n--;
		}
	}
	return n;
}
```