

Proiect Programare Orientata Obiect

Hoka David-Stelian

Semigrupa 2

Proiectul contine:

- o clasa generica cu rol de lista ce aloca dinamic mai multe elemente de un anumit tip, de exemplu int, string sau chiar clase;
- clasa lista contine teste pentru fiecare functionalitate rulate prin intermediul bibliotecii "assert"
- clasa de baza Persoana;
- clasa Student, ce mosteneste clasa Persoana;
- clasa Disciplina;
- clasa Profesor, ce mosteneste clasa Student si inregistreaza maxim 5 discipline per profesor;
- clasa ui (user interface) pentru interfata grafica ce imi afiseaza un meniu de comenzi.
- header "Service.h" in care am realizat cele doua functionalitati de relatiune intre liste si anumite functii pentru rularea meniului.

Ideea de ansamblu al proiectului este de a realiza o lista alocata dinamic pentru studenti, profesori si discipline cu optiunile de adaugare, modificare sau stergere element din lista. Nu in ultimul rand, am creat urmatoarele relatiuni intre liste:

- 1) afisarea disciplinelor predate de fiecare profesor in parte;
- 2) afisarea profesorilor si a studentilor in functie de facultate (Ex: Profesorul x predă lui y).

Capturi de ecran cu initializarea fiecarii clase:

VectorDinamic.h:

```

Service.h      main.cpp     ui.h       Profesor.h   Student.h   VectorDinamic.h   Disciplina.h   Teste.h     Persoana.h
☒ probare project → lista<TipElem>
1  #pragma once
2  #include "Persoana.h"
3  #include "Student.h"
4  #include <assert.h>
5  #include <iostream>
6  #include <cstdlib>
7  template <class TipElem>
8  class lista {
9
10    TipElem* elems = NULL; //elementele listei
11    int lg; //lungime
12    int cap; //capacitate
13
14 public:
15    lista(TipElem*, int, int); //constructor cu parametri - (1)
16    lista() {}; //constructor fara parametri - (2)
17    ~lista(); // desctuctor - (3)
18    lista(const lista&); //constructor de copiere - (4)
19    void creare_lista(); //creare lista goala - (5)
20    TipElem* get(int); //returnare element (6)
21    void set(int, TipElem); //modifica element (7)
22    int* size(); //returneaza numarul de elemente din lista (8)
23    void capacitate_asigurata(); //asigura capacitatea listei (9)
24    void add(TipElem); //adauga student in lista (10)
25    void copiere_lista(lista*); //copiaza "superficial" lista (11)
26    TipElem& retur_elems(); //returneaza elementele (12)
27    void sterge_element(int); //sterge element de pe pozitie data (13)
28    template<typename TipElem> friend istream& operator>>(istream&, lista&); //citeste o lista in flux (14)
29    template<typename TipElem> friend ostream& operator<<(ostream&, lista); //afiseaza o lista in flux(15)
30    void genereaza_listă1(); //genereaza o lista de 10 elemente (16)
31    void afisare(); //afiseaza lista de elemente (17)
32    int& retur_lg(); //returneaza lungimea listei (18)

```

```

Service.h      main.cpp     ui.h       Profesor.h   Student.h   VectorDinamic.h   Disciplina.h   Teste.h     Persoana.h
☒ probare project → lista<TipElem>
23    void add(TipElem); //adauga student in lista (10)
24    void copiere_lista(lista*); //copiaza "superficial" lista (11)
25    TipElem& retur_elems(); //returneaza elementele (12)
26    void sterge_element(int); //sterge element de pe pozitie data (13)
27    template<typename TipElem> friend istream& operator>>(istream&, lista&); //citeste o lista in flux (14)
28    template<typename TipElem> friend ostream& operator<<(ostream&, lista); //afiseaza o lista in flux(15)
29    void genereaza_listă1(); //genereaza o lista de 10 elemente (16)
30    void afisare(); //afiseaza lista de elemente (17)
31    int& retur_lg(); //returneaza lungimea listei (18)
32    int& retur_cap(); //returneaza capacitatea listei (19)
33    TipElem& retur_elems(int); //returneaza elementele listei(20)
34    TipElem& operator[](int); //operator indexare (21)
35    void genereaza_listă2(); //genereaza o lista de 5 elemente
36    void genereaza_listă3(lista<Profesor>); //genereaza o lista de 30 de materii (10 materii per facultate)
37  };
38  //TESTARE - 8 functii
39  template <class TipElem>
40  void test_creeare_lista(); //testarea crearii unei liste (22)
41  void test_iterare_lista(); //testarea iterarii listei (23)
42  void test_copiere_lista(); //testarea copierii listei (24)
43  void test_resize(); //testarea redimensionarii unei liste (25)
44  void test_stergere(); //testarea stergerii unui element (26)
45  void test_modificare_element(); // testarea modificarii elementelor (27)
46  void test_generare(); //testarea generarii a 10 elemente (28)
47  void testare_vectorDinamic(); //apelarea testelor de mai sus (29)
48
49  template <class TipElem>
50  inline lista<TipElem>::lista(TipElem* el, int x, int y) : lg(x), cap(y) {
51    elems = new TipElem[cap];
52  }
53

```

Persoana.h:

```
Service.h      main.cpp      ui.h       Profesor.h      Student.h      VectorDi
☒ probare project
1     #pragma once
2     //Persona.h
3     #include<string>
4     #include<iostream>
5     using namespace std;
6     class Persoana
7     {
8     protected:
9         char* nume = NULL;
10        int varsta;
11    public: /** constructor cu parametri*/
12        Persoana(const char* nume, int varsta = 0);
13        Persoana(char nume, int varsta = 0);
14        /** Constructor de copiere*/
15        Persoana(const Persoana& p);
16        Persoana() {};
17        ~Persoana();
18        char& getNume(); //returneaza numele
19        int& getVarsta(); //returneaza varsta
20        Persoana& operator = (Persoana& p);
21        friend istream& operator>>(istream&, Persoana&);
22        friend ostream& operator<<(ostream&, Persoana);
23
24    };
25
26    Persoana::Persoana(const char* nume, int varsta)
27    {
28        this->nume = new char[strlen(nume) + 1];
29        strcpy(this->nume, nume);
30        this->varsta = varsta;
31        //cout << "Apel constr. Persoana\n";
109 %  □ 0  ▢ 1  ← →
```

Student.h:

The screenshot shows a code editor window with the tab bar at the top containing files: Service.h, main.cpp, ui.h, Profesor.h, Student.h (selected), VectorDinamic.h, Disciplina.h, Teste.h, and Persoana.h. Below the tabs is a search bar labeled "probare project" and "(Global Scope)". The main area displays the code for the Student class:

```
1 #pragma once
2 #include "Persoana.h"
3 class Student : public Persoana
4 {
5 protected:
6     char* facultate = NULL;
7 public:
8     Student(const char* nume, int varsta, const char* facultate); //cu const
9     Student(Persoana p, const char* facultate);
10    Student(const Student& s);
11    Student() {};
12    ~Student();
13    Student get_student(); //copiază student
14    void set_student(Student);
15    Student& retur();
16    char& retur_nume();
17    char& retur_facultate();
18    int& retur_varsta();
19    friend istream& operator>>(istream&, Student&);
20    friend ostream& operator<<(ostream&, Student);
21    //void afisare();
22    Student& operator = (Student& p);
23 };
24
25 Student::Student(const char* nume, int varsta, const char* facultate) :Persoana(nume, varsta)
26 {
27     this->facultate = new char[strlen(facultate) + 1];
28     strcpy(this->facultate, facultate);
29     //cout << "Apel constr. Student\n";
30 }
```

At the bottom left, there is a zoom level indicator "109%" and a status bar message "No issues found".

Disciplina.h:

The screenshot shows a code editor window with the tab bar at the top containing Service.h, main.cpp, ui.h, Profesor.h, Student.h, VectorDinamic.h, and Disciplina.h. The Disciplina.h tab is selected, indicated by a blue border. Below the tabs is a toolbar with icons for file operations. The main area displays the source code for the Disciplina class:

```
1 #pragma once
2 #include <string>
3 #include "Profesor.h"
4 #include <algorithm>
5
6 class Disciplina {
7     string cod;
8     string nume;
9 public:
10    Disciplina(string, string);
11    Disciplina() {};
12    Disciplina(const Disciplina&);
13    ~Disciplina();
14    string& retur_DiscNume();
15    string getCod() const;
16    friend ostream& operator<<(ostream&, Disciplina);
17    friend istream& operator>>(istream&, Disciplina&);
18 };
19
20 Disciplina::Disciplina(string x, string y) : cod(x), nume(y) {
21     std::transform(nume.begin(), nume.end(), nume.begin(), ::toupper);
22 }
23
24 Disciplina::Disciplina(const Disciplina& d) : cod(d.cod), nume(d.nume) {
25     std::transform(nume.begin(), nume.end(), nume.begin(), ::toupper);
26 }
27
28 Disciplina::~Disciplina() {}
29
30 string& Disciplina::retur_DiscNume() {
31     return nume;
```

At the bottom left, there is a status bar showing "109 %". At the bottom right, there is a message "No issues found" with a green checkmark icon.

Profesor.h:

Service.h main.cpp ui.h Profesor.h ✘ Student.h VectorDinamic.h Disciplina.h Teste.h Persoana.h

```

probare project → Profesor
1 #pragma once
2 #include "Student.h"
3 #include <iostream>
4 #include <string>
5 #include <algorithm>
6 #include "Disciplina.h"
7 #define MAX_nrDiscPredate 5
8 using namespace std;
9
10 class Profesor : public Student {
11     int nrDiscPredate = 0;
12     Disciplina* discPredate[MAX_nrDiscPredate];
13 public:
14     Profesor(const char*, int, const char*);
15     Profesor();
16     ~Profesor();
17     void addDiscPredate(Disciplina*);
18     Disciplina& getDisciplina(int) const;
19     void afisare_disciplini();
20     string& getNumDisc();
21     int& getNrDisc();
22     string getCodDisc();
23     Profesor& citire_prof_materii();
24     friend ostream& operator<<(ostream&, Profesor);
25     void afisare_nume(int);
26 };
27
28 Profesor::Profesor(const char* nume, int varsta, const char* facultate) : Student(nume, varsta, facultate) {}
29
30 int& Profesor::getNrDisc() {

```

109% 0 1 ← →

ui.h:

Service.h main.cpp ui.h Profesor.h Student.h VectorDinamic.h Disciplina.h Teste.h Persoana.h

```

probare project (Global Scope)
1 #pragma once
2 #include <iostream>
3 #include <string>
4 using namespace std;
5 class ui {
6     string interfata; //interfata principală
7     string interfata_afiseaza; //interfata secundară pentru afisare
8     string interfata_adauga; //interfata secundară pentru adăugare
9     string interfata_modifica; //interfata sec. pt. modificare
10    string interfata_sterge; //interfata sec. pt. stergere
11 public:
12     ui(char*, char*, char*, char*); //constructor implicit
13     ui();
14     ui(const ui&); //constructorul de copiere
15     ~ui(); //destructor
16     void creare_interfata(); //creare interfata pentru programul implicit
17     void afisare_interfata_principală(); //afisare interfata principală
18     void afisare_interf_adauga(); //afisare interf. de adăugare
19     void afisare_interf_afiseaza(); //afisare interf. de afisare
20     void afisare_interf_modifica(); //afisare interf. de modificare
21     void afisare_interf_sterge(); //afisare interf. de stergere
22 };
23
24 ui::ui(char* x, char* y, char* z, char* a, char* b) : interfata1(x), interfata_afiseaza(y), interfata_adauga(z), interfata_modifica(a), interfata_sterge(b) {} //constructor implicit
25
26
27 ui::ui(const ui& ui) { //constructorul de copiere
28     interfata1 = ui.interfata;
29     interfata_adauga = ui.interfata_adauga;
30     interfata_afiseaza = ui.interfata_afiseaza;
31     interfata_modifica = ui.interfata_modifica;
32     interfata_sterge = ui.interfata_sterge;
33 }

```

109% 0 No issues found

Service.h:

Service.h main.cpp ui.h Profesor.h Student.h VectorDinamic.h Disciplina.h Teste.h

probare project (Global Scope)

```

1 #pragma once
2 #include <iostream>
3 #include <string>
4 #include <algorithm>
5 #include "Persoana.h"
6 #include "Student.h"
7 #include "Profesor.h"
8 #include "Disciplina.h"
9 #include "ui.h"
10 #include "Teste.h"
11 #include "VectorDinamic.h"
12 using namespace std;
13 void atribuire_materii_generalizate(lista<Profesor>& lp, lista<Disciplina>& ld) {
14     //STIINTE
15     for (int i = 0; i < 5; i++)
16         lp.get(0)->addDiscPredare(ld.get(i));
17     for (int i = 5; i < 10; i++)
18         lp.get(3)->addDiscPredare(ld.get(i));
19     //MEDICINA
20     for (int i = 10; i < 15; i++)
21         lp.get(2)->addDiscPredare(ld.get(i));
22     for (int i = 15; i < 20; i++)
23         lp.get(5)->addDiscPredare(ld.get(i));
24     //SPORT
25     for (int i = 20; i < 25; i++)
26         lp.get(1)->addDiscPredare(ld.get(i));
27     for (int i = 25; i < 30; i++)
28         lp.get(4)->addDiscPredare(ld.get(i));
29 }
30 
```

109 % No issues found

Service.h main.cpp ui.h Profesor.h Student.h VectorDinamic.h Disciplina.h Teste.h Persoana.h

probare project (Global Scope)

```

30
31 void afisare_Prof_Studenti(lista<Profesor>* lp, lista<Student>* ls) {
32     for (int i = 0; i < *lp->size(); i++) {
33         cout << "\n\nProfesorul " << &lp->get(i)->getNum() << " preda la facultatea de " << &lp->get(i)->return_facultate() << " si ii are ca studenti pe:\n";
34         for (int j = 0; j < *ls->size(); j++) {
35             if (strcmp(&ls->get(j)->return_facultate(), &lp->get(i)->return_facultate()) == 0)
36                 cout << " | " << &ls->get(j)->getNum() << " | ";
37         }
38     }
39 }
40 cout << endl;
41 }
42
43 void afisare_Prof_materii(lista<Profesor> lp, lista<Disciplina> ld) {
44     for (int i = 0; i < lp.return_size(); i++)
45         lp.get(i)->afisare_disciplini();
46 }
47
48
49 void meniu_afisare(int cmd, lista<Student>& ls, lista<Profesor>& lp, lista<Disciplina>& ld) {
50     switch (cmd) {
51     case 1:
52         cout << "\nPantru studenti\n\n";
53         ls.afisare();
54         break;
55     case 2:
56         cout << "\nPantru profesori\n\n";
57         lp.afisare();
58         break;
59     case 3:
60     }
61 }
62 
```

109 % No issues found

main.cpp:

Service.h main.cpp ui.h Profesor.h Student.h VectorDinamic.h

probare project (Global)

```
1 #define _CRTDBG_MAP_ALLOC
2 #include <crtdbg.h>
3 #include <string>
4 #include "Persoana.h"
5 #include "Student.h"
6 #include "Profesor.h"
7 #include "Disciplina.h"
8 #include "ui.h"
9 #include "Teste.h"
10 #include "VectorDinamic.h"
11 #include "Service.h"
12
13 int main()
14 {
15     test_all();
16     cout << "TESTARE COMPLETA!\n\n";
17     lista<Student> ls;
18     //cin >> ls;
19     ls.genereaza_listă1(); //generez lista studenti (10)
20     //ls.afisare();
21     lista<Profesor> lp;
22     lp.genereaza_listă2(); //generez lista profesori (5)
23     //lp.afisare();
24     lista<Disciplina> ld;
25     ld.genereaza_listă3(lp); //generez lista discipline (30)
26     //ld.afisare();
27     /*atribuire_materii_generalizate(lp, ld);
28     for (int i = 0; i < *lp.size(); i++)
29         lp.get(i)->afisare_disciplini();*/
30 }
```

109 % No issues found

Service.h main.cpp ui.h Profesor.h Student.h VectorDinamic.h

(Global)

```
22     lista<Profesor> lp;
23     lp.genereaza_listă2(); //generez lista profesori (5)
24     //lp.afisare();
25     lista<Disciplina> ld;
26     ld.genereaza_listă3(lp); //generez lista discipline (30)
27     //ld.afisare();
28     /*atribuire_materii_generalizate(lp, ld);
29     for (int i = 0; i < *lp.size(); i++)
30         lp.get(i)->afisare_disciplini();*/
31     atribuire_materii_generalizate(lp, ld);
32     ui ui;
33     ui.creare_interfata();
34     //ui.afisare_interfata_principala();
35     //atribuireProfesorStudenti(&lp, &ls);
36     int cmd = -1, cm2 = 1;
37     while (cmd != 5) {
38         ui.afisare_interfata_principala();
39         cout << "Dati comanda: ";
40         cin >> cmd;
41         switch (cmd) {
42             case 1:
43                 ui.afisare_interf_afiseaza();
44                 cout << "Efectuati comanda: ";
45                 cin >> cm2;
46                 validare_comanda_afisare(cm2);
47                 meniu_afisare(cm2, ls, lp, ld);
48                 break;
49             case 2:
50                 ui.afisare_interf_adauga();
51                 cout << "Efectuati comanda: ";
```

109 % No issues found

Service.h main.cpp ui.h Profesor.h Student.h VectorDinami

probare project

```
49         case 2:
50             ui.afisare_interf_adauga();
51             cout << "Efectuati comanda: ";
52             cin >> cm2;
53             validare_comanda_general(cm2);
54             meniu_adaugare(cm2, ls, lp, ld);
55             break;
56         case 3:
57             ui.afisare_interf_modifica();
58             cout << "Efectuati comanda: ";
59             cin >> cm2;
60             validare_comanda_general(cm2);
61             meniu_modificare(cm2, ls, lp, ld);
62             break;
63         case 4:
64             ui.afisare_interf_sterge();
65             cout << "Efectuati comanda: ";
66             cin >> cm2;
67             validare_comanda_general(cm2);
68             meniu_stergere(cm2, ls, lp, ld);
69             break;
70         case 5:
71             cout << "Program inchis cu succes.\n";
72             break;
73         default:
74             cout << "Comanda invalida!\n";
75             break;
76     }
77 }
78 }
```

109 % No issues found

```
70     case 5:
71         cout << "Program inchis cu succes.\n";
72         break;
73     default:
74         cout << "Comanda invalida!\n";
75         break;
76     }
77 }
78 }
79 }
80 _CrtDumpMemoryLeaks();
81 return (0);
82 }
```

109 % No issues found

Rularea programului:

AFISARI:

```
C:\Users\ACASA\Desktop\probare proiect
TEST CREARE LISTA COMPLET!
TEST ITERARE LISTA COMPLET!
TEST COPIERE LISTA COMPLET!
TEST STERGE ELEMENT COMPLET!
TEST MODIFICA ELEMNT COMPLET!
TEST RESIZE LISTA COMPLET!
TEST GENERARE LISTA COMPLET!
TESTARE COMPLETA!

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda:
```

```
C:\Users\ACASA\Desktop\probare proiect\Debug\probare.exe

TEST CREARE LISTA COMPLET!
TEST ITERARE LISTA COMPLET!
TEST COPIERE LISTA COMPLET!
TEST STERGE ELEMENT COMPLET!
TEST MODIFICA ELEMNT COMPLET!
TEST RESIZE LISTA COMPLET!
TEST GENERARE LISTA COMPLET!
TESTARE COMPLETA!

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: 1
    1 - Afiseaza Student
    2 - Afiseaza Profesor
    3 - Afiseaza materie
    4 - Afiseaza Profesor-materii
    5 - Afiseaza Profesor-studenti
    6 - Inapoi
Efectuati comanda: -
```

```
C:\Users\ACASA\Desktop\probare proiect\Debug\probare proiect.exe
    2 - Afiseaza Profesor
    3 - Afiseaza materie
    4 - Afiseaza Profesor-materii
    5 - Afiseaza Profesor-studenti
    6 - Inapoi
Efectuati comanda: 1

Pentru studenti

Lista este:
ID: 0 - Nume: Andrei Varsta: 29 Facultate: Stiinte
ID: 1 - Nume: David Varsta: 18 Facultate: Medicina
ID: 2 - Nume: Florina Varsta: 29 Facultate: Sport
ID: 3 - Nume: Maria Varsta: 23 Facultate: Stiinte
ID: 4 - Nume: Sebastian Varsta: 21 Facultate: Medicina
ID: 5 - Nume: George Varsta: 28 Facultate: Sport
ID: 6 - Nume: Mihai Varsta: 23 Facultate: Stiinte
ID: 7 - Nume: Florin Varsta: 26 Facultate: Medicina
ID: 8 - Nume: Mark Varsta: 22 Facultate: Sport
ID: 9 - Nume: Petru Varsta: 20 Facultate: Stiinte

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda:
```

```
C:\Users\ACASA\Desktop\probare proiect\Debug\probare proiect.exe
4 - Sterge element
5 - Exit
Dati comanda: 1
    1 - Afiseaza Student
    2 - Afiseaza Profesor
    3 - Afiseaza materie
    4 - Afiseaza Profesor-materii
    5 - Afiseaza Profesor-studenti
    6 - Inapoi
Efectuati comanda: 2

Pentru profesori

Lista este:
ID: 0 - Profesor - Nume: Marian | Varsta: 35 | Facultate: Stiinte
ID: 1 - Profesor - Nume: Ionescu | Varsta: 50 | Facultate: Medicina
ID: 2 - Profesor - Nume: Popescu | Varsta: 55 | Facultate: Sport
ID: 3 - Profesor - Nume: Anamaria | Varsta: 47 | Facultate: Stiinte
ID: 4 - Profesor - Nume: Daniela | Varsta: 57 | Facultate: Medicina
ID: 5 - Profesor - Nume: Monica | Varsta: 64 | Facultate: Sport

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: 
```

```
C:\Users\ACASA\Desktop\probare proiect\Debug\probare proiect.exe
```

```
Efectuati comanda: 3
```

```
Pentru discipline
```

```
Lista este:
```

```
ID: 0 - Cod: 760x1 Nume disciplina: OOP
ID: 1 - Cod: sls7e Nume disciplina: FP
ID: 2 - Cod: 9th4l Nume disciplina: SPD
ID: 3 - Cod: pcv7u Nume disciplina: ANALIZA
ID: 4 - Cod: vh32x Nume disciplina: ALGEBRA
ID: 5 - Cod: 6k7vh Nume disciplina: GEOMETRIE
ID: 6 - Cod: lvzwe Nume disciplina: BAZE DE DATE
ID: 7 - Cod: logak Nume disciplina: SO
ID: 8 - Cod: eoeeo Nume disciplina: WEB PROG
ID: 9 - Cod: jo78o Nume disciplina: INTELIGENTA ARTIFICIALA
ID: 10 - Cod: 47vuj Nume disciplina: FOTBAL
ID: 11 - Cod: xaka6 Nume disciplina: BASKET
ID: 12 - Cod: 8692y Nume disciplina: INOT
ID: 13 - Cod: dct3k Nume disciplina: AEROBIC
ID: 14 - Cod: 047rj Nume disciplina: HANDBAL
ID: 15 - Cod: l54y4 Nume disciplina: VOLEI
ID: 16 - Cod: jc34r Nume disciplina: BOX
ID: 17 - Cod: r62g7 Nume disciplina: DANS
ID: 18 - Cod: 3vvyk Nume disciplina: RALIU
ID: 19 - Cod: gry99 Nume disciplina: VANATOARE SUBACVATICA
ID: 20 - Cod: dw920 Nume disciplina: BIOLOGIA CELULARA
ID: 21 - Cod: 8vq8k Nume disciplina: BIOFIZICA
ID: 22 - Cod: dxh7n Nume disciplina: BIOCHIMIE
ID: 23 - Cod: l7pfr Nume disciplina: BIOTECA
ID: 24 - Cod: zc3mo Nume disciplina: ANATOMIE
ID: 25 - Cod: 8c87i Nume disciplina: CHIRURGIE
ID: 26 - Cod: alg9w Nume disciplina: PSIHOLOGIA
ID: 27 - Cod: xmrud Nume disciplina: HISTOLOGIE
ID: 28 - Cod: tjco5 Nume disciplina: GENETICA
ID: 29 - Cod: uoeyj Nume disciplina: MICROBIOLOGIE
```

```
MENIU COMENZI
```

```
Prelucrari cu liste
```

- 1 - Afisare lista
- 2 - Adauga in lista
- 3 - Modifica element
- 4 - Sterge element
- 5 - Exit

```
Dati comanda: -
```

```
Efectuati comanda: 4
```

```
Afisare Profesori-materii predate
```

```
Profesorul Marian predă: OOP, FP, SPD, ANALIZA, ALGEBRA.
```

```
Profesorul Ionescu predă: BIOLOGIA CELULARA, BIOFIZICA, BIOCHIMIE, BIOTECA, ANATOMIE.
```

```
Profesorul Popescu predă: FOTBAL, BASKET, INOT, AEROBIC, HANDBAL.
```

```
Profesorul Anamaria predă: GEOMETRIE, BAZE DE DATE, SO, WEB PROG, INTELIGENTA ARTIFICIALA.
```

```
Profesorul Daniela predă: CHIRURGIE, PSIHOLOGIA, HISTOLOGIE, GENETICA, MICROBIOLOGIE.
```

```
Profesorul Monica predă: VOLEI, BOX, DANS, RALIU, VANATOARE SUBACVATICA.
```

```
MENIU COMENZI
```

```
Prelucrari cu liste
```

- 1 - Afisare lista
- 2 - Adauga in lista
- 3 - Modifica element
- 4 - Sterge element
- 5 - Exit

```
Dati comanda: _
```

```
Efectuati comanda: 5
```

```
Afisare Profesor-Studenti
```

```
Profesorul Marian predă la facultatea de Stiinte si ii are ca studenti pe:  
| Andrei | | Maria | | Mihai | | Petru |
```

```
Profesorul Ionescu predă la facultatea de Medicina si ii are ca studenti pe:  
| David | | Sebastian | | Florin |
```

```
Profesorul Popescu predă la facultatea de Sport si ii are ca studenti pe:  
| Florina | | George | | Mark |
```

```
Profesorul Anamaria predă la facultatea de Stiinte si ii are ca studenti pe:  
| Andrei | | Maria | | Mihai | | Petru |
```

```
Profesorul Daniela predă la facultatea de Medicina si ii are ca studenti pe:  
| David | | Sebastian | | Florin |
```

```
Profesorul Monica predă la facultatea de Sport si ii are ca studenti pe:  
| Florina | | George | | Mark |
```

```
MENIU COMENZI
```

```
Prelucrari cu liste
```

- 1 - Afisare lista
- 2 - Adauga in lista
- 3 - Modifica element
- 4 - Sterge element
- 5 - Exit

```
Dati comanda:
```

ADAUGARI:

```
C:\Users\ACASA\Desktop\probare project\Debug\probare project.exe
Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: 2
    1 - Adauga Student
    2 - Adauga Profesor
    3 - Adauga materie
    4 - Adauga Profesor-materii
    5 - Inapoi
Efectuati comanda: 1
Citire student
Nume: vasile
Varsta: 23
Facultatea: medicina

Student adaugat cu succes!
Studentul este: Nume: Vasile | Varsta: 23 | Facultate: Medicina

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda:
```

```
C:\Users\ACASA\Desktop\probare project\Debug\probare project.exe
Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: 2
    1 - Adauga Student
    2 - Adauga Profesor
    3 - Adauga materie
    4 - Adauga Profesor-materii
    5 - Inapoi
Efectuati comanda: 2
Citire profesor
Nume: gheorghe
Varsta: 38
Facultatea: stiinte

Profesor adaugat cu succes!
Profesorul este: Profesor - Nume: Gheorghe | Varsta: 38 | Facultate: Stiinte

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda:
```

```
C:\Users\ACASA\Desktop\probare proiect\Debug\probare proiect.exe
MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: 2
    1 - Adauga Student
    2 - Adauga Profesor
    3 - Adauga materie
    4 - Adauga Profesor-materii
    5 - Inapoi
Efectuati comanda: 3
Citire disciplina
Codul generat: ypmeb Nume disciplina: istorie

Disciplina adaugata cu succes!
Disciplina este: Cod: ypmeb Nume disciplina: ISTORIE

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda:
```

```
C:\Users\ACASA\Desktop\probare proiect\Debug\probare proiect.exe
4 - Sterge element
5 - Exit
Dati comanda: 2
    1 - Adauga Student
    2 - Adauga Profesor
    3 - Adauga materie
    4 - Adauga Profesor-materii
    5 - Inapoi
Efectuati comanda: 4
Citire Profesor-materii
Nume: ion
Varsta: 33
Facultatea: stiinte
Numar materii: 2

Pentru profesorul Ion se citesc 2 materii
Codul generat: ypmeb Nume disciplina: java
Codul generat: srt4c Nume disciplina: c#

Profesorul Ion predă: JAVA, C#.

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: 
```

MODIFICARI:

```
C:\Users\ACASA\Desktop\probare proiect\Debug\probare proiect.exe
Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: 3
    1 - Modifica Student
    2 - Modifica Profesor
    3 - Modifica materie
    4 - Modifica materie dupa ID prof
    5 - Inapoi
Efectuati comanda: 1
Modifica studentul cu ID-ul: 0
Studentul: Nume: Andrei | Varsta: 29 | Facultate: Stiinte
Date noi
Nume: catalin
Varsta: 21
Facultatea: sport
Dupa modificare: Nume: Catalin | Varsta: 21 | Facultate: Sport

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda:
```

```
C:\Users\ACASA\Desktop\probare project\Debug\probare project.exe
Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: 3
    1 - Modifica Student
    2 - Modifica Profesor
    3 - Modifica materie
    4 - Modifica materie dupa ID prof
    5 - Inapoi
Efectuati comanda: 2
Modifica proful cu ID-ul: 0
Profesorul: Profesor - Nume: Marian | Varsta: 35 | Facultate: Stiinte
Date noi
Nume: Tudor
Varsta: 37
Facultatea: stiinte
Dupa modificare: Profesor - Nume: Tudor | Varsta: 37 | Facultate: Stiinte
MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda:
```

```
C:\Users\ACASA\Desktop\probare proiect\Debug\probare proiect.exe
MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: 3
    1 - Modifica Student
    2 - Modifica Profesor
    3 - Modifica materie
    4 - Modifica materie dupa ID prof
    5 - Inapoi
Efectuati comanda: 3
Modifica disciplina cu ID-ul: 0
Disciplina: Cod: 760xl Nume disciplina: OOP
Date noi
Codul generat: ypmeb Nume disciplina: unity
Dupa modificarare: Cod: ypmeb Nume disciplina: UNITY

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda:
```

```
C:\Users\ACASA\Desktop\probare project\Debug\probare project.exe
Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: 3
    1 - Modifica Student
    2 - Modifica Profesor
    3 - Modifica materie
    4 - Modifica materie dupa ID prof
    5 - Inapoi
Efectuati comanda: 4
ID prof: 4
Profesorul Daniela predă: CHIRURGIE, PSIHOLOGIA, HISTOLOGIE, GENETICA, MICROBIOLOGIE.

Modificati materia (nume): genetica
Nume nou: bioteca
Dupa modificarile efectuate:
Profesorul Daniela predă: CHIRURGIE, PSIHOLOGIA, HISTOLOGIE, BIOTECNA, MICROBIOLOGIE.

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: -
```

STERGERI:

```
Dati comanda: 4
    1 - Sterge Student
    2 - Sterge Profesor
    3 - Sterge materie
    4 - Sterge Profesor-materii
    5 - Inapoi
Efектuati comanda: 1
Sterge studentul cu ID-ul: 3
Studentul ce va fi sters: Nume: Maria | Varsta: 23 | Facultate: Stiinte
Element sters cu succes.
MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda:
```

DUPA STERGERE STUDENT:

```
C:\Users\ACASA\Desktop\probare project\Debug\probare project.exe

1 - Afiseaza Student
2 - Afiseaza Profesor
3 - Afiseaza materie
4 - Afiseaza Profesor-materii
5 - Afiseaza Profesor-studenti
6 - Inapoi
Efectuati comanda: 1

Pentru studenti

Lista este:
ID: 0 - Nume: Andrei | Varsta: 29 | Facultate: Stiinte
ID: 1 - Nume: David | Varsta: 18 | Facultate: Medicina
ID: 2 - Nume: Florina | Varsta: 29 | Facultate: Sport
ID: 3 - Nume: Sebastian | Varsta: 21 | Facultate: Medicina
ID: 4 - Nume: George | Varsta: 28 | Facultate: Sport
ID: 5 - Nume: Mihai | Varsta: 23 | Facultate: Stiinte
ID: 6 - Nume: Florin | Varsta: 26 | Facultate: Medicina
ID: 7 - Nume: Mark | Varsta: 22 | Facultate: Sport
ID: 8 - Nume: Petru | Varsta: 20 | Facultate: Stiinte

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda:
```

```
Dati comanda: 4
    1 - Sterge Student
    2 - Sterge Profesor
    3 - Sterge materie
    4 - Sterge Profesor-materii
    5 - Inapoi
Efектуати comanda: 2
Sterge proful cu ID-ul: 2
Profesorul ce va fi sters: Profesor - Nume: Popescu | Varsta: 55 | Facultate: Sport
Element sters cu succes.
MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: _
```

DUPA STERGERE PROFESOR:

```
Efectuati comanda: 2

Pentru profesori

Lista este:
ID: 0 - Profesor - Nume: Marian | Varsta: 35 | Facultate: Stiinte
ID: 1 - Profesor - Nume: Ionescu | Varsta: 50 | Facultate: Medicina
ID: 2 - Profesor - Nume: Anamaria | Varsta: 47 | Facultate: Stiinte
ID: 3 - Profesor - Nume: Daniela | Varsta: 57 | Facultate: Medicina
ID: 4 - Profesor - Nume: Monica | Varsta: 64 | Facultate: Sport

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: -
```

```
Dati comanda: 4
    1 - Sterge Student
    2 - Sterge Profesor
    3 - Sterge materie
    4 - Sterge Profesor-materii
    5 - Inapoi
Efектуати comanda: 3
Sterge disciplina cu ID-ul: 4
Disciplina ce va fi stearsa: Cod: vh32x Nume disciplina: ALGEBRA
Element sters cu succes.

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: -
```

DUPA STERGERE DISCIPLINA:

```
C:\Users\ACASA\Desktop\probare project\Debug\probare project.exe
    6 - Inapoi
Efektuati comanda: 3

Pentru discipline

Lista este:
ID: 0 - Cod: 760xl Nume disciplina: OOP
ID: 1 - Cod: sls7e Nume disciplina: FP
ID: 2 - Cod: 9th4l Nume disciplina: SPD
ID: 3 - Cod: pcv7u Nume disciplina: ANALIZA
ID: 4 - Cod: 6k7vh Nume disciplina: GEOMETRIE
ID: 5 - Cod: lvzwe Nume disciplina: BAZE DE DATE
ID: 6 - Cod: logak Nume disciplina: SO
ID: 7 - Cod: eoeeo Nume disciplina: WEB PROG
ID: 8 - Cod: jo78o Nume disciplina: INTELIGENTA ARTIFICIALA
ID: 9 - Cod: 47vuj Nume disciplina: FOTBAL
ID: 10 - Cod: xaka6 Nume disciplina: BASKET
ID: 11 - Cod: 8692y Nume disciplina: INOT
ID: 12 - Cod: dct3k Nume disciplina: AEROBIC
ID: 13 - Cod: 047rj Nume disciplina: HANDBAL
ID: 14 - Cod: 154y4 Nume disciplina: VOLEI
ID: 15 - Cod: jc34r Nume disciplina: BOX
ID: 16 - Cod: r62g7 Nume disciplina: DANS
ID: 17 - Cod: 3vvyk Nume disciplina: RALIU
ID: 18 - Cod: gry99 Nume disciplina: VANATOARE SUBACVATICA
ID: 19 - Cod: dw920 Nume disciplina: BIOLOGIA CELULARA
ID: 20 - Cod: 8vq8k Nume disciplina: BIOFIZICA
ID: 21 - Cod: dxh7n Nume disciplina: BIOCHIMIE
ID: 22 - Cod: l7pfr Nume disciplina: BIOTECA
ID: 23 - Cod: zc3mo Nume disciplina: ANATOMIE
ID: 24 - Cod: 8c87i Nume disciplina: CHIRURGIE
ID: 25 - Cod: alg9w Nume disciplina: PSIHOLOGIA
ID: 26 - Cod: xmrud Nume disciplina: HISTOLOGIE
ID: 27 - Cod: tjco5 Nume disciplina: GENETICA
ID: 28 - Cod: uoeyj Nume disciplina: MICROBIOLOGIE

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: ■
```

```
C:\Users\ACASA\Desktop\probare proiect\Debug\probare proiect.exe
MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: 4
    1 - Sterge Student
    2 - Sterge Profesor
    3 - Sterge materie
    4 - Sterge Profesor-materii
    5 - Inapoi
Efectuati comanda: 4
Sterge Profesor-materii
ID prof: 0
Se va sterge Profesorul: Marian si se vor sterge materiile urmatoare:
Profesorul Marian predă: OOP, FP, SPD, ANALIZA, ALGEBRA.

Elementele au fost eliminate cu succes.
MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: -
```

DUPA STERGERE PROF – MATERII:

```
Pentru profesori

Lista este:
ID: 0 - Profesor - Nume: Ionescu | Varsta: 50 | Facultate: Medicina
ID: 1 - Profesor - Nume: Popescu | Varsta: 55 | Facultate: Sport
ID: 2 - Profesor - Nume: Anamaria | Varsta: 47 | Facultate: Stiinte
ID: 3 - Profesor - Nume: Daniela | Varsta: 57 | Facultate: Medicina
ID: 4 - Profesor - Nume: Monica | Varsta: 64 | Facultate: Sport

MENIU COMENZI

Prelucrari cu liste
1 - Afisare lista
2 - Adauga in lista
3 - Modifica element
4 - Sterge element
5 - Exit
Dati comanda: -
```

```
Pentru discipline
```

```
Lista este:
```

```
ID: 0 - Cod: 6k7vh Nume disciplina: GEOMETRIE
ID: 1 - Cod: lvzwe Nume disciplina: BAZE DE DATE
ID: 2 - Cod: logak Nume disciplina: SO
ID: 3 - Cod: eoeeo Nume disciplina: WEB PROG
ID: 4 - Cod: jo78o Nume disciplina: INTELIGENTA ARTIFICIALA
ID: 5 - Cod: 47vuj Nume disciplina: FOTBAL
ID: 6 - Cod: xaka6 Nume disciplina: BASKET
ID: 7 - Cod: 8692y Nume disciplina: INOT
ID: 8 - Cod: dct3k Nume disciplina: AEROBIC
ID: 9 - Cod: 047rj Nume disciplina: HANDBAL
ID: 10 - Cod: 154y4 Nume disciplina: VOLEI
ID: 11 - Cod: jc34r Nume disciplina: BOX
ID: 12 - Cod: r62g7 Nume disciplina: DANS
ID: 13 - Cod: 3vvyk Nume disciplina: RALIU
ID: 14 - Cod: gry99 Nume disciplina: VANATOARE SUBACVATICA
ID: 15 - Cod: dw920 Nume disciplina: BIOLOGIA CELULARA
ID: 16 - Cod: 8vq8k Nume disciplina: BIOFIZICA
ID: 17 - Cod: dxh7n Nume disciplina: BIOCHIMIE
ID: 18 - Cod: 17pfr Nume disciplina: BIOTECA
ID: 19 - Cod: zc3mo Nume disciplina: ANATOMIE
ID: 20 - Cod: 8c87i Nume disciplina: CHIRURGIE
ID: 21 - Cod: alg9w Nume disciplina: PSIHOLOGIA
ID: 22 - Cod: xmrud Nume disciplina: HISTOLOGIE
ID: 23 - Cod: tjco5 Nume disciplina: GENETICA
ID: 24 - Cod: uoeyj Nume disciplina: MICROBIOLOGIE
```

```
MENIU COMENZI
```

```
Prelucrari cu liste
```

- 1 - Afisare lista
- 2 - Adauga in lista
- 3 - Modifica element
- 4 - Sterge element
- 5 - Exit

```
Dati comanda:
```

CODURILE:

PERSOANA.H:

```
#pragma once
```

```
//Persona.h
```

```

#include<string>
#include<iostream>
#include <algorithm>
using namespace std;

class Persoana

{
protected:
    char* nume = NULL;
    int varsta;

public: /** constructor cu parametri*/
    Persoana(const char* nume, int varsta = 0);
    Persoana(char nume, int varsta = 0);
    /** Constructor de copiere*/
    Persoana(const Persoana& p);
    Persoana() {};
    ~Persoana();
    char& getNume(); //returneaza numele
    int& getVarsta(); //returneaza varsta
    Persoana& operator = (Persoana& p);
    friend istream& operator>>(istream&, Persoana&);
    friend ostream& operator<<(ostream&, Persoana);
};

Persoana::Persoana(const char* nume, int varsta)

```

```

{

    this->nume = new char[strlen(nume) + 1];
    strcpy(this->nume, nume);
    this->varsta = varsta;
    //cout << "Apel constr. Persoana\n";

}

Persoana::Persoana(char nume, int varsta) {

    const char* lung_s = &nume;

    this->nume = new char[strlen(lung_s) + 1];
    strcpy(this->nume, lung_s);
    this->varsta = varsta;

}

Persoana::Persoana(const Persoana& p)

{

    nume = new char[strlen(p.nume) + 1];
    strcpy(nume, p.nume); varsta = p.varsta;
    //cout << "Apel Constr. de copiere Persoana\n";

}

Persoana::~Persoana() {

    //if(nume)
    delete nume;
    varsta = -1;
    //cout << "Apel destructor Persoana\n";

}

```

```

char& Persoana::getNume() {
    return *nume;
}

int& Persoana::getVarsta() {
    return varsta;
}

Persoana& Persoana::operator = (Persoana& p)
{
    //cout << "Apel atribuire Persoana\n";
    if (this != &p)
    {
        nume = new char[strlen(p.nume) + 1];
        strcpy(nume, p.nume);
        varsta = p.varsta;
    }
    return *this;
}

istream& operator>>(istream& x, Persoana& p) {
    cout << "Nume: ";
    char nume[20];
    x >> nume;
}

```

```

nume[0] = toupper(nume[0]);

cout << "Varsta: ";

int varsta=0;

x >> varsta;

while (varsta < 18) {

    cout << "Varsta trebuie sa fie minim 18: ";

    x >> varsta;

}

Persoana p2(nume, varsta);

p = p2;

return x;

}

```

```

ostream& operator<<(ostream& x, Persoana p) {

x << "Nume: " << p.nume << " | Varsta: " << p.varsta << " ";

return x;

}

```

STUDENT.H:

```

#pragma once

#include "Persoana.h"

class Student : public Persoana

{

protected:

    char* facultate = NULL;

```

```

public:
    Student(const char* nume, int varsta, const char* facultate); //cu const
    Student(Persoana p, const char* facultate);
    Student(const Student& s);
    Student() {};
    ~Student();
    Student get_student(); //copiază student
    void set_student(Student);
    Student& retur();
    char& retur_nume();
    char& retur_facultate();
    int& retur_varsta();
    friend istream& operator>>(istream&, Student&);
    friend ostream& operator<<(ostream&, Student);
    //void afisare();
    Student& operator = (Student& p);
};


```

```

Student::Student(const char* nume, int varsta, const char* facultate) :Persoana(nume, varsta)
{
    this->facultate = new char[strlen(facultate) + 1];
    strcpy(this->facultate, facultate);
    //cout << "Apel constr. Student\n";
}


```

```
Student::Student(Persoana p, const char* facultate) : Persoana(p) {  
    this->facultate = new char[strlen(facultate) + 1];  
    strcpy(this->facultate, facultate);  
}
```

```
Student::Student(const Student& s) : Persoana(s.nume, s.varsta)
```

```
{  
    facultate = new char[strlen(s.facultate) + 1];  
    strcpy(facultate, s.facultate);  
    //cout << "Apel Constr. de copiere Student\n";  
}
```

```
Student::~Student()
```

```
{  
    delete facultate;  
    //cout << "Apel destructor Student\n";  
}
```

```
void Student::set_student(Student s) {
```

```
    this->~Student();
```

```
    *this = s;
```

```
}
```

```
Student& Student::return() {
```

```
    return *this;
```

```
}
```

```
Student Student::get_student() {
```

```
    return *this;
```

```
}
```

```
char& Student::retur_nume() {
```

```
    return *nume;
```

```
}
```

```
char& Student::retur_facultate() {
```

```
    return *facultate;
```

```
}
```

```
int& Student::retur_varsta() {
```

```
    return varsta;
```

```
}
```

```
istream& operator>>(istream& x, Student& s) {
```

```
    Persoana p;
```

```
    cin >> p;
```

```
    s.nume = &p.getNume();
```

```
    s.nume = new char[strlen(&p.getNume()) + 1];
```

```
    strcpy(s.nume, &p.getNume());
```

```
    s.varsta = p.getVarsta();
```

```

char facultate[20];

cout << "Facultatea: ";

x >> facultate;

facultate[0] = toupper(facultate[0]);

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

strcpy(s.facultate, facultate);

return x;

}

```

```

ostream& operator<<(ostream& x, Student s) {

Persoana p(s.nume, s.varsta);

x << p;

x << " | Facultate: " << s.facultate << '\n';

return x;

}

```

```

Student& Student::operator = (Student& s)

{

//cout << "Apel atribuire Student\n";

if (this != &s)

{

    Persoana::operator=(s);

    //delete[]facultate;

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

    strcpy(facultate, s.facultate);
}

```

```
    }

    return *this;

}
```

PROFESOR.H:

```
#pragma once

#include "Student.h"

#include <iostream>

#include <string>

#include <algorithm>

#include "Disciplina.h"

#define MAX_nrDiscPredate 5

using namespace std;

class Profesor : public Student {

    int nrDiscPredate = 0;

    Disciplina* discPredate[MAX_nrDiscPredate];

public:

    Profesor(const char*,int,const char*);

    Profesor() {};

    ~Profesor() { *discPredate = NULL; nrDiscPredate = 0; };

    void addDiscPredate(Disciplina*);

    Disciplina& getDisciplina(int) const;

    void afisare_disciplini();

    string& getNumeDisc(int);
```

```

int& getNrDisc();

string getCodDisc(int);

Profesor& citire_prof_materii();

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

void afisare_nume(int);

};

Profesor::Profesor(const char* nume, int varsta, const char* facultate) : Student(nume, varsta,
facultate) {}

int& Profesor::getNrDisc() {

    return nrDiscPredate;

}

string Profesor::getCodDisc(int x) {

    return discPredate[x]->getCod();

}

string& Profesor::getNumeDisc(int i) {

    std::transform(discPredate[i]->retur_DiscNume().begin(),
discPredate[i]->retur_DiscNume().end(), discPredate[i]->retur_DiscNume().begin(), ::toupper);

    return discPredate[i]->retur_DiscNume();

}

void Profesor::addDiscPredate(Disciplina* disciplina) {

```

```
discPredate[nrDiscPredate++] = disciplina;  
}
```

```
Disciplina& Profesor::getDisciplina(int i) const {  
    //cout << &discPredate[i] << endl;  
    return *discPredate[i];  
}
```

```
void Profesor::afisare_disciplini() {  
    cout << "Profesorul " << &this->getNume() << " preda: ";  
    for (int i = 0; i < this->nrDiscPredate; i++) {  
        string nd = this->getNumeDisc(i);  
        cout << nd;  
        if (i < this->nrDiscPredate - 1)  
            cout << ", ";  
        else cout << ".\n";  
    }  
    cout << '\n';  
}
```

```
Profesor& citire_prof_materii() {  
    Profesor p;  
    cin >> p;  
    cout << "Numar materii: ";  
    int nr;
```

```

    cin >> nr;

    while (nr < 1 || nr>5) {

        cout << "Numar materii: ";

        cin >> nr;

    }

    Disciplina d;

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

        cin >> d;

        p.addDiscPredare(&d);

    }

    return p;

}

```

```

ostream& operator<<(ostream& x, Profesor p) {

    const char* fac = &p.retur_facultate();

    Student s = Student(Persoana(&p.getNume(), p.getVarsta()), fac);

    x << "Profesor - " << s;

    return x;

}

```

DISCIPLINA.H:

```

#pragma once

#include <string>

#include "Profesor.h"

#include <algorithm>

```

```

class Disciplina {
    string cod;
    string nume;
public:
    Disciplina(string, string);
    Disciplina() {};
    Disciplina(const Disciplina&);

    ~Disciplina();

    string& retur_DiscNume();

    string getCod() const;

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

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

};

Disciplina::Disciplina(string x, string y) : cod(x), nume(y) {
    std::transform(nume.begin(), nume.end(), nume.begin(), ::toupper);
}

Disciplina::Disciplina(const Disciplina& d) : cod(d.cod), nume(d.nume) {
    std::transform(nume.begin(), nume.end(), nume.begin(), ::toupper);
}

Disciplina::~Disciplina() {}

```

```

string& Disciplina::retur_DiscNume() {
    return nume;
}

}

string Disciplina::getCod()const {
    return cod;
}

ostream& operator<<(ostream& x, Disciplina d) {
    x << "Cod: " << d.cod << " Nume disciplina: " << d.nume << '\n';
    return x;
}

istream& operator>>(istream& x, Disciplina& d) {
    cout << "Codul generat: ";
    string txt = "abcdefghijklmnopqrstuvwxyz0123456789";
    char cod[6] = "";
    for (int i = 0; i < 5; i++)
        cod[i] = txt[rand() % 36];
    d.cod = cod;
    cout << cod;
    cout << " Nume disciplina: ";
    string nume;
    x >> nume;
    d.nume = nume;
}

```

```
    return x;  
}  
  
}
```

VectorDinamic.H:

```
#pragma once  
  
#include "Persoana.h"  
  
#include "Student.h"  
  
#include <assert.h>  
  
#include <string>  
  
#include <cstdlib>  
  
template <class TipElem>  
  
class lista {  
  
    TipElem* elems = NULL; //elementele listei  
  
    int lg; //lungime  
  
    int cap; //capacitate  
  
public:  
  
    lista(TipElem*, int, int); //constructor cu parametri - (1)  
  
    lista() {}; //constructor fara parametri - (2)  
  
    ~lista(); // desctuctor - (3)  
  
    lista(const lista&); //constructor de copiere - (4)  
  
    void creare_lista(); //creare lista goala - (5)  
  
    TipElem* get(int); //returnare element (6)  
  
    void set(int, TipElem); //modifica element (7)  
  
    int* size(); //returneaza numarul de elemente din lista (8)
```

```

void capacitate_asigurata(); //asigura capacitatea listei (9)

void add(TipElem); //adauga student in lista (10)

void copiere_lista(lista*); //copiaza "superficial" lista (11)

TipElem& retur_elems(); //returneaza elementele (12)

void sterge_element(int); //sterge element de pe pozitie data (13)

template<typename TipElem> friend istream& operator>>(istream&, lista&); //citeste o lista in
flux (14)

template<typename TipElem> friend ostream& operator<<(ostream&, lista); //afiseaza o lista in
flux(15)

void genereaza_lista1(); //genereaza o lista de 10 elemente (16)

void afisare(); //afiseaza lista de elemente (17)

int& retur_lg(); //returneaza lungimea listei (18)

int& retur_cap(); //returneaza capacitatea listei (19)

TipElem& retur_elems(int); //returneaza elementele listei(20)

TipElem& operator[](int); //operator indexare (21)

void genereaza_lista2(); //genereaza o lista de 5 elemente (22)

void genereaza_lista3(lista<Profesor>&); //genereaza o lista de 30 de materii (10 materii per
facultate) (23)

};

//TESTARE - 8 functii

template <class TipElem>

void test_creare_lista(); //testarea crearii unei liste (24)

void test_iterare_lista(); //testarea iterarii listei (25)

void test_copiere_lista(); //testarea copierii listei (26)

void test_resize(); //testarea redimensionarii unei liste (27)

void test_stergere(); //testarea stergerii unui element (28)

void test_modificare_element(); // testarea modificarii elementelor (29)

```

```
void test_generare(); //testarea generarii a 10 elemente (30)  
void testare_vectorDinamic(); //apelarea testelor de mai sus (31)
```

```
template <class TipElem>  
  
inline lista<TipElem>::lista(TipElem* el, int x, int y) : lg(x), cap(y) {  
  
    elems = new TipElem[cap];  
  
}
```

```
template <class TipElem>  
  
lista<TipElem>::~lista() {  
  
    //for (int i = 0; i < lg; i++)  
  
    //elems[i].~Student();  
  
    delete[] elems;  
  
    elems = NULL;  
  
    lg = -1;  
  
    cap = -1;  
  
    //cout << "Apel desctuctor lista\n";  
  
}
```

```
template <class TipElem>  
  
TipElem& lista<TipElem>::operator[](int i) {  
  
    return elems[i];  
  
}
```

```
template <class TipElem>
```

```

inline lista<TipElem>::lista(const lista& l) {

    lg = l.lg;
    cap = l.cap;
    for (int i = 1; i <= l.lg; i++)
        elems[i] = l.elems[i];
}

```

```

template <class TipElem>

void lista<TipElem>::creare_lista() {

    lg = 0;
    cap = 2;
    elems = new TipElem[cap];
}

```

```

template <class TipElem>

TipElem* lista<TipElem>::get(int poz) {
    return &elems[poz];
}

```

```

template <class TipElem>

void lista<TipElem>::set(int poz, TipElem s) {
    TipElem* cp = new TipElem[cap];
    int k = 0;
    for (int i = 0; i < lg; i++)
        if (i != poz)

```

```

        cp[k++] = elems[i];

    delete[]elems;

    cp[poz] = s;

    elems = cp;

}

```

```

template <class TipElem>

int* lista<TipElem>::size() {

    return &lg;

}

```

```

template <class TipElem>

void lista<TipElem>::capacitate_asigurata() {

    if (lg == cap) {

        //mai aloca memorie

        int capacitate_noua = cap + 2;

        TipElem* nElems = new TipElem[capacitate_noua];

        //copiază elemente

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

            nElems[i] = elems[i];

        delete[] elems; //dealoca vectorul vechi

        elems = nElems;

        cap = capacitate_noua;

    }

}

```

```

template <class TipElem>

void lista<TipElem>::add(TipElem el) {
    this->capacitate_asigurata();
    elems[lg++] = el;
}

template <class TipElem>

void lista<TipElem>::copiere_lista(lista* l) {
    this->creare_lista();
    for (int i = 0; i < *l->size(); i++) {
        this->add(*l->get(i));
    }
}

template <class TipElem>
TipElem& lista<TipElem>::retur_elems() {
    return *this->elems;
}

template <class TipElem>
void lista<TipElem>::sterge_element(int poz) {
    TipElem* cp = new TipElem[cap];
    int k = 0;
    for (int i = 0; i < lg; i++)

```

```

        if (i != poz)

            cp[k++] = elems[i];

        delete[] elems;

        elems = cp;

        lg--;

    }

template <class TipElem>

istream& operator>>(istream& x, lista<TipElem>& l) {

    l.creare_lista();

    int lungime;

    cout << "Lungimea listei: ";

    x >> lungime;

    TipElem t;

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

        x >> t;

        l.add(t);

    }

    return x;

}

```

```

template <class TipElem>

ostream& operator<<(ostream& x, lista<TipElem> l) {

    x << "Lista este:\n";

    for (int i = 0; i < l.retur_lg(); i++) {

```

```

        x << "ID: " << i << " - " << l[i];
    }

    x << endl;

    return x;
}

template<class TipElem>

int& lista<TipElem>::retur_lg() {

    return this->lg;
}

template<class TipElem>

int& lista<TipElem>::retur_cap() {

    return this->cap;
}

template<class TipElem>

TipElem& lista<TipElem>::retur_elems(int poz) {

    return this->elems[poz];
}

template <class TipElem>

void lista<TipElem>::genereaza_lista1() {

    char nume_sir[100] = "Andrei,David,Florina,Maria,Sebastian,George,Mihai,Florin,Mark,Petru";
    const char virgula = ',';
}

```

```

char facultate1[20] = "Stiinte";
char facultate2[20] = "Medicina";
char facultate3[20] = "Sport";
int Max, Min;
Max = 30;
Min = 18;
int varsta = rand() % (Max + 1 - Min) + Min;
char nume[10] = "";
int start = 0;
for (int i = start; i < strlen(nume_sir); i++) {
    const char litera = nume_sir[i];
    if (litera == virgula) {
        start = i + 1;
        i = strlen(nume_sir);
    }
    else
        nume[i] = nume_sir[i];
}
this->add(TipElem(nume, varsta, facultate1));
for (int i = 1; i < 10; i++) {
    varsta = rand() % (Max + 1 - Min) + Min;
    char nume[20] = "";
    int dim_nume = 0;
    for (int k = start; k < strlen(nume_sir); k++) {
        const char litera = nume_sir[k];

```

```

        if (litera == virgula) {

            start = k + 1;

            k = strlen(nume_sir);

        }

        else

            nume[dim_nume++] = nume_sir[k];

    }

    if (strcmp(&this->get(i - 1)->retur_facultate(), facultate1) == 0)

        this->add(TipElem(nume, varsta, facultate2));

    else

        if (strcmp(&this->get(i - 1)->retur_facultate(), facultate2) == 0)

            this->add(TipElem(nume, varsta, facultate3));

        else

            this->add(TipElem(nume, varsta, facultate1));

    }

}

template <class TipElem>

void lista<TipElem>::afisare() {

    cout << "Lista este:\n";

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

        cout << "ID: " << i << " - " << this->retur_elems(i);

    }

    cout << endl;

}

```

```

template <class TipElem>

void lista<TipElem>::genereaza_lista2() {

    char nume_sir[80] = "Marian,Ionescu,Popescu,Anamaria,Daniela,Monica";
    const char virgula = ',';
    char facultate1[20] = "Stiinte";
    char facultate2[20] = "Medicina";
    char facultate3[20] = "Sport";
    int Max, Min;
    Max = 65;
    Min = 23;
    int varsta = rand() % (Max + 1 - Min) + Min;
    char nume[10] = "";
    int start = 0;
    for (int i = start; i < strlen(nume_sir); i++) {
        const char litera = nume_sir[i];
        if (litera == virgula) {
            start = i + 1;
            i = strlen(nume_sir);
        }
        else
            nume[i] = nume_sir[i];
    }
    this->add(TipElem(nume, varsta, facultate1));
    for (int i = 1; i < 6; i++) {

```

```

varsta = rand() % (Max + 1 - Min) + Min;

char nume[20] = "";

int dim_nume = 0;

for (int k = start; k < strlen(nume_sir); k++) {

    const char litera = nume_sir[k];

    if (litera == virgula) {

        start = k + 1;

        k = strlen(nume_sir);

    }

    else

        nume[dim_nume++] = nume_sir[k];

}

if (strcmp(&this->get(i - 1)->retur_facultate(), facultate1) == 0)

    this->add(TipElem(nume, varsta, facultate2));

else

    if (strcmp(&this->get(i - 1)->retur_facultate(), facultate2) == 0)

        this->add(TipElem(nume, varsta, facultate3));

    else

        this->add(TipElem(nume, varsta, facultate1));

}

}

template <class TipElem>

void lista<TipElem>::genereaza_lista3(lista<Profesor>& lp) {

    char nume_sir[281] = "OOP,FP,SPD,ANALIZA,ALGEBRA,GEOMETRIE,BAZE DE DATE,SO,WEB
PROG,INTELIGENTA

```

ARTIFICIALA,FOTBAL,BASKET,INOT,AEROBIC,HANDBAL,VOLEI,BOX,DANS,RALIU,VANATOARE
SUBACVATICA,BIOLOGIA
CELULARA,BIOFIZICA,BIOCHIMIE,BIOTECA,ANATOMIE,CHIRURGIE,PSIHOLOGIA,HISTOLOGIE,GENETICA,M
ICROBIOLOGIE,";

```
const char virgula = ',';

string txt = "";

txt = "abcdefghijklmnopqrstuvwxyz0123456789";

char cod[6] = "";

for (int i = 0; i < 5; i++)

    cod[i] = txt[rand() % 36];

char nume[100] = "";

int start = 0;

for (int i = start; i < strlen(nume_sir); i++) {

    const char litera = nume_sir[i];

    if (litera == virgula) {

        start = i + 1;

        i = strlen(nume_sir);

    }

    else

        nume[i] = nume_sir[i];

}

this->add(Disciplina(cod, nume));

for (int i = 1; i < 30; i++) {

    char cod[6] = "";

    for (int i = 0; i < 5; i++)

        cod[i] = txt[rand() % 36];

    char nume[100] = "";
```

```

int dim_nume = 0;

for (int k = start; k < strlen(nume_sir); k++) {

    const char litera = nume_sir[k];

    if (litera == virgula) {

        start = k + 1;

        k = strlen(nume_sir);

    }

    else

        nume[dim_nume++] = nume_sir[k];

}

this->add(Disciplina(cod, nume));

}

}

void test_creare_lista() {

    lista<Student> l;

    l.creare_lista();

    l.^lista();

    assert(*l.size() == -1);

    cout << "TEST CREARE LISTA COMPLET!\n";

}

void test_iterare_lista() {

    lista<Student> l;

```

```

l.creare_lista();

l.add(Student("a", 20, "b"));

l.add(Student("a2", 21, "b2"));

assert(*l.size() == 2);

Student* s = l.get(0);

assert(strcmp(&s->retur_nume(), "a") == 0);

s->set_student(*l.get(1));

assert(strcmp(&s->retur_facultate(), "b2") == 0);

cout << "TEST ITERARE LISTA COMPLET!\n";

}

```

```

void test_copiere_lista() {

lista<Student> l;

l.creare_lista();

l.add(Student("a", 19, "b"));

l.add(Student("c", 20, "d"));

lista<Student> l2;

l2.copiere_lista(&l);

assert(*l2.size() == 2);

Student s = *l2.get(0);

assert(strcmp(&s.retur_nume(), "a") == 0);

l.^lista();

l2.^lista();

```

```
    cout << "TEST COPIERE LISTA COMPLET!\n";
}

}
```

```
void test_resize() {
    lista<Student> l;
    l.creare_lista();
    for (int i = 0; i < 10; i++) {
        l.add(Student("a", 20, "b"));
    }
    assert(*l.size() == 10);
    l.^lista();
    assert(*l.size() == -1);
    assert(&l.retur_elems() == NULL);
    cout << "TEST RESIZE LISTA COMPLET!\n";
}

}
```

```
void test_stergere() {
    lista<Student> l;
    l.creare_lista();
    l.add(Student("a", 19, "a"));
    l.add(Student("b", 20, "b"));
    l.add(Student("c", 21, "c"));
    l.sterge_element(1);
}
```

```

    assert(*l.size() == 2);

    assert(strcmp(&l.get(0)->retur_nume(), "a") == 0);

    assert(strcmp(&l.get(1)->retur_nume(), "c") == 0);

    cout << "TEST STERGE ELEMENT COMPLET!\n";

}


```

```

void test_modificare_element() {

    lista<Student> l;

    l.creare_lista();

    l.add(Student("a", 19, "a"));

    Student s = Student("b", 22, "b");

    l.set(0, s);

    assert(strcmp(&l.get(0)->retur_nume(), "b") == 0);

    cout << "TEST MODIFICA ELEMNT COMPLET!\n";

}


```

```

void test_generare() {

    lista<Student> l;

    l.creare_lista();

    l.genereaza_lista1();

    assert(*l.size() == 10);

    assert(strcmp(&l.get(0)->retur_nume(), "Andrei") == 0);

    cout << "TEST GENERARE LISTA COMPLET!\n";

}


```

```
}
```



```
void testare_vectorDinamic() {
```

```
    test_creeare_lista();
```

```
    test_iterare_lista();
```

```
    test_copiere_lista();
```

```
    test_stergere();
```

```
    test_modificare_element();
```

```
    test_resize();
```

```
    test_generare();
```

```
}
```

TESTE.H:

```
#pragma once
```

```
//#include "Disciplina.h"
```

```
#include "Persoana.h"
```

```
#include "Student.h"
```

```
#include "VectorDinamic.h"
```



```
void test_all() {
```

```
    testare_vectorDinamic();
```

```
}
```

UI.H:

```
#pragma once
```

```

#include <iostream>

#include <string>

using namespace std;

class ui {

    string interfata1; //interfata principala

    string interfata_afiseaza; //interfata secundara pentru afisare

    string interfata_adauga; //interfata secundara pentru adaugare

    string interfata_modifica; //interfata sec. pt. modificare

    string interfata_sterge; //interfata sec. pt. stergere

public:

    ui(char*, char*,char*,char*,char*); //construtor implicit

    ui() {};

    ui(const ui&); //constructorul de copiere

    ~ui() {};//destructor

    void creare_interfata(); //creare interfata pentru programul implicit

    void afisare_interfata_principala(); //afisare interfata principala

    void afisare_interf_adauga(); //afisare interf. de adaugare

    void afisare_interf_afiseaza(); //afisare interf. de afisare

    void afisare_interf_modifica(); //afisare interf. de modificare

    void afisare_interf_sterge(); //afisare interf. de stergere

};

ui::ui(char* x, char* y, char* z, char* a, char* b) : interfata1(x), interfata_afiseaza(y),
interfata_adauga(z), interfata_modifica(a), interfata_sterge(b) //construtor implicit

}

```

```
ui::ui(const ui& ui) { //constructorul de copiere  
    interfata1 = ui.interfata1;  
  
    interfata_adauga = ui.interfata_adauga;  
  
    interfata_afiseaza = ui.interfata_afiseaza;  
  
    interfata_modifica = ui.interfata_modifica;  
  
    interfata_sterge = ui.interfata_sterge;  
}
```

```
void ui::creare_interfata() { //creare interfata pentru programul implicit  
  
    interfata1 = "MENIU COMENZI\n\n";  
  
    interfata1 += "Prelucrari cu liste\n";  
  
    interfata1 += "1 - Afisare lista\n";  
  
    interfata1 += "2 - Adauga in lista\n";  
  
    interfata1 += "3 - Modifica element\n";  
  
    interfata1 += "4 - Sterge element\n";  
  
    interfata1 += "5 - Exit\n";  
  
  
    interfata_adauga += "\t1 - Adauga Student\n";  
  
    interfata_adauga += "\t2 - Adauga Profesor\n";  
  
    interfata_adauga += "\t3 - Adauga materie\n";  
  
    interfata_adauga += "\t4 - Adauga Profesor-materii\n";  
  
    interfata_adauga += "\t5 - Inapoi\n";  
  
  
    interfata_afiseaza += "\t1 - Afiseaza Student\n";  
    interfata_afiseaza += "\t2 - Afiseaza Profesor\n";
```

```

interfata_afiseaza += "\t3 - Afiseaza materie\n";
interfata_afiseaza += "\t4 - Afiseaza Profesor-materii\n";
interfata_afiseaza += "\t5 - Afiseaza Profesor-studenti\n";
interfata_afiseaza += "\t6 - Inapoi\n";

interfata_modifica += "\t1 - Modifica Student\n";
interfata_modifica += "\t2 - Modifica Profesor\n";
interfata_modifica += "\t3 - Modifica materie\n";
interfata_modifica += "\t4 - Modifica materie dupa ID prof\n";
interfata_modifica += "\t5 - Inapoi\n";

interfata_sterge += "\t1 - Sterge Student\n";
interfata_sterge += "\t2 - Sterge Profesor\n";
interfata_sterge += "\t3 - Sterge materie\n";
interfata_sterge += "\t4 - Sterge Profesor-materii\n";
interfata_sterge += "\t5 - Inapoi\n";

}

void ui::afisare_interfata_principala() { //afisare interfata principala
    cout << interfata1;
}

void ui::afisare_interf_adauga() { //afisare interf. de adaugare
    cout << interfata_adauga;
}

```

```

}

void ui::afisare_interf_afiseaza() { //afisare interf. de afisare
    cout << interfata_afiseaza;
}

void ui::afisare_interf_modifica() { //afisare interf. de modificare
    cout << interfata_modifica;
}

void ui::afisare_interf_sterge() { //afisare interf. de stergere
    cout << interfata_sterge;
}

void validare_comanda_general(int& cmd) {
    if (cmd != 1 && cmd != 2 && cmd != 3 && cmd != 4 && cmd != 5) {
        cout << "Comanda invalida!\n";
        cout << "Redare comanda: ";
        cin >> cmd;
    }
}

void validare_comanda_afisare(int& cmd) {
    if (cmd != 1 && cmd != 2 && cmd != 3 && cmd != 4 && cmd != 5 && cmd != 6) {
        cout << "Comanda invalida!\n";
        cout << "Redare comanda: ";
        cin >> cmd;
    }
}

```

```
}
```

SERVICE.H:

```
#pragma once

#include <iostream>

#include <string>

#include <algorithm>

#include "Persoana.h"

#include "Student.h"

#include "Profesor.h"

#include "Disciplina.h"

#include "ui.h"

#include "Teste.h"

#include "VectorDinamic.h"

using namespace std;

void atribuire_materii_generalizate(lista<Profesor>& lp, lista<Disciplina>& ld) {

    //STIINTE

    for (int i = 0; i < 5; i++)

        lp.get(0)->addDiscPredare(ld.get(i));

    for (int i = 5; i < 10; i++)

        lp.get(3)->addDiscPredare(ld.get(i));

    //MEDICINA

    for (int i = 10; i < 15; i++)

        lp.get(2)->addDiscPredare(ld.get(i));

    for (int i = 15; i < 20; i++)
}
```

```

    lp.get(5)->addDiscPredate(Id.get(i));

    //SPORT

    for (int i = 20; i < 25; i++)
        lp.get(1)->addDiscPredate(Id.get(i));

    for (int i = 25; i < 30; i++)
        lp.get(4)->addDiscPredate(Id.get(i));

    }

void afisare_Profesor_Studenti(lista<Profesor>* lp, lista<Student>* ls) {
    for (int i = 0; i < *lp->size(); i++) {
        cout << "\n\nProfesorul " << &lp->get(i)->getNum() << " preda la facultatea de " <<
        &lp->get(i)->retur_facultate() << " si ii are ca studenti pe:\n";
        for (int j = 0; j < *ls->size(); j++) {
            if (strcmp(&ls->get(j)->retur_facultate(), &lp->get(i)->retur_facultate()) == 0) {
                cout << " | " << &ls->get(j)->getNum() << " | ";
            }
        }
    }
    cout << endl;
}

void afisare_prof_materii(lista<Profesor> lp, lista<Disciplina> Id) {
    for (int i = 0; i < lp.retur_lg(); i++)
        lp.get(i)->afisare_disciplini();
}

```

```

void sterge_disciplina(string s, lista<Disciplina> &Id) {
    for (int i = 0; i < *Id.size(); i++)
        if (Id.get(i)->retur_DiscNume() == s)
            Id.sterge_element(i);
}

void meniu_afisare(int cmd, lista<Student>& ls, lista<Profesor>& lp, lista<Disciplina>& id) {
    switch (cmd) {
        case 1:
            cout << "\nPentru studenti\n\n";
            ls.afisare();
            break;
        case 2:
            cout << "\nPentru profesori\n\n";
            lp.afisare();
            break;
        case 3:
            cout << "\nPentru discipline\n\n";
            id.afisare();
            break;
        case 4:
            cout << "\nAfisare Profesori-materii predate\n\n";
            for (int i = 0; i < *lp.size(); i++) {
                lp.retur_elems(i).afisare_disciplini();
            }
    }
}

```

```

        break;

case 5:

    cout << "\nAfisare Profesor-Studenti\n\n";
    afisare_Profesor_Studenti(&lp, &ls);
    break;

case 6:

    break;

default:

    cout << "Comanda invalida!\n\n";
    break;
}

}

void meniu_adaugare(int cmd, lista<Student>& ls, lista<Profesor>& lp, lista<Disciplina>& ld) {

    Student s; Profesor p; Disciplina d;
    int nr = NULL;

    switch (cmd) {

        case 1:

            cout << "Citire student\n";

            cin >> s;

            ls.add(s);

            cout << "\nStudent adaugat cu succes!\nStudentul este: " << *ls.get(*ls.size() - 1) << '\n';

            break;

        case 2:

            cout << "Citire profesor\n";

```

```

    cin >> p;

    lp.add(p);

    cout << "\nProfesor adaugat cu succes!\nProfesorul este: " << *lp.get(*lp.size() - 1) <<
    '\n';

    break;

case 3:

    cout << "Citire disciplina\n";

    cin >> d;

    ld.add(d);

    cout << "\nDisciplina adaugata cu succes!\nDisciplina este: " << *ld.get(*ld.size() - 1) <<
    '\n';

    break;

case 4:

    cout << "Citire Profesor-materii\n";

    cin >> p;

    cout << "Numar materii: ";

    cin >> nr;

    cout << "\nPentru profesorul " << &p.getNume() << " se citesc " << nr << " materii\n";

    lp.add(p);

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

        cin >> d;

        ld.add(d);

        lp.get(*lp.size()-1)->addDiscPredare(ld.get(*ld.size()-1));

    }

    cout << endl;

    lp.get(*lp.size() - 1)->afisare_disciplini();

```

```

        break;

case 5:
    break;

default:
    cout << "Comanda invalida!\n";
    break;
}
}
```

```

void meniu_modificare(int cmd, lista<Student>& ls, lista<Profesor>& lp, lista<Disciplina>& ld) {
    Student s; Profesor p; Disciplina d;
    int id = NULL;
    string nume, nume2;
    switch (cmd) {
        case 1:
            cout << "Modifica studentul cu ID-ul: ";
            cin >> id;
            cout << "Studentul: ";
            cout << *ls.get(id);
            cout << "Date noi\n";
            cin >> s;
            ls.set(id, s);
            cout << "Dupa modificare: ";
            cout << *ls.get(id)<<'\n';
            break;
    }
}
```

case 2:

```
cout << "Modifica proful cu ID-ul: ";
cin >> id;
cout << "Profesorul: ";
cout << *Ip.get(id);
cout << "Date noi\n";
cin >> p;
Ip.set(id, p);
cout << "Dupa modificare: ";
cout << *Ip.get(id) << '\n';
break;
```

case 3:

```
cout << "Modifica disciplina cu ID-ul: ";
cin >> id;
cout << "Disciplina: ";
cout << *Id.get(id);
cout << "Date noi\n";
cin >> d;
Id.set(id, d);
cout << "Dupa modificare: ";
cout << *Id.get(id) << '\n';
break;
```

case 4:

```
cout << "ID prof: ";
cin >> id;
```

```

lp.get(id)->afisare_disciplini();

cout << "Modificati materia (nume): ";

cin >> nume;

std::transform(nume.begin(), nume.end(), nume.begin(), ::toupper);

for(int i=0;i<lp.get(id)->getNrDisc();i++)

    if (lp.get(id)->getNumeDisc(i) == nume) {

        cout << "Nume nou: ";

        cin >> nume2;

        lp.get(id)->getNumeDisc(i) = nume2;

    }

    cout << "Dupa modificare:\n";

    lp.get(id)->afisare_disciplini();

    break;

case 5:

    break;

default:

    cout << "Comanda invalida!\n";

    break;

}

void meniu_stergere(int cmd, lista<Student>& ls, lista<Profesor>& lp, lista<Disciplina>& ld) {

    int poz = NULL,nrdisc,i,j,size;

    string s[5];

    switch (cmd) {

```

case 1:

```
cout << "Sterge studentul cu ID-ul: ";
cin >> poz;
cout << "Studentul ce va fi sters: " << *ls.get(poz);
ls.sterge_element(poz);
cout << "Element sters cu succes.\n";
break;
```

case 2:

```
cout << "Sterge proful cu ID-ul: ";
cin >> poz;
cout << "Profesorul ce va fi sters: " << *lp.get(poz);
lp.sterge_element(poz);
cout << "Element sters cu succes.\n";
break;
```

case 3:

```
cout << "Sterge disciplina cu ID-ul: ";
cin >> poz;
cout << "Disciplina ce va fi stearsa: " << *ld.get(poz);
ld.sterge_element(poz);
cout << "Element sters cu succes.\n";
break;
```

case 4:

```
cout << "Sterge Profesor-materii\n";
cout << "ID prof: ";
cin >> poz;
```

```

cout << "Se va sterge Profesorul: " << &lp.get(poz)->getNume() << " si se vor sterge
materiile urmatoare:\n";

lp.get(poz)->afisare_disciplini();

nrdisc = lp.get(poz)->getNrDisc();

for (i = 0; i < nrdisc; i++)

    s[i] = lp.get(poz)->getDisciplina(i).getCod();

    j = nrdisc-1;

    size = *ld.size()-1;

while (j >= 0) {

    for (i = 0; i < size; i++) {

        if (ld.get(i)->getCod() == s[j]) {

            ld.sterge_element(i);

        }

    }

    j--;

    size--;

}

lp.get(poz)->getNrDisc() = 0;

lp.sterge_element(poz);

cout << "Elementele au fost eliminate cu succes.\n";

break;

case 5:

break;

default:

```

```

        cout << "Comanda invalida!\n";
        break;
    }
}

```

MAIN.CPP:

```

#define _CRTDBG_MAP_ALLOC

#include <crtdbg.h>

#include <string>

#include "Persoana.h"

#include "Student.h"

#include "Profesor.h"

#include "Disciplina.h"

#include "ui.h"

#include "Teste.h"

#include "VectorDinamic.h"

#include "Service.h"

int main()

{
    test_all();

    cout << "TESTARE COMPLETA!\n\n";

    lista<Student> ls;

    //cin >> ls;
}

```

```

ls.genereaza_lista1(); //generez lista studenti (10)

//ls.afisare();

lista<Profesor> lp;

lp.genereaza_lista2(); //generez lista profesori (5)

//lp.afisare();

lista<Disciplina> ld;

ld.genereaza_lista3(lp); //generez lista discipline (30)

//ld.afisare();

/*atribuire_materii_generalizate(lp, ld);

for (int i = 0; i < *lp.size(); i++)

    lp.get(i)->afisare_disciplini();*/
}

atribuire_materii_generalizate(lp, ld);

ui ui;

ui.creare_interfata();

//ui.afisare_interfata_principala();

//atribuireProfesorStudenti(&lp, &ls);

int cmd = -1, cm2 = 1;

while (cmd != 5) {

    ui.afisare_interfata_principala();

    cout << "Dati comanda: ";

    cin >> cmd;

    switch (cmd) {

        case 1:

            ui.afisare_interf_afiseaza();

            cout << "Efectuati comanda: ";

```

```

    cin >> cm2;

    validare_comanda_afisare(cm2);

    meniu_afisare(cm2, ls, lp, ld);

    break;

case 2:

    ui.afisare_interf_adauga();

    cout << "Efectuati comanda: ";

    cin >> cm2;

    validare_comanda_general(cm2);

    meniu_adaugare(cm2, ls, lp, ld);

    break;

case 3:

    ui.afisare_interf_modifica();

    cout << "Efectuati comanda: ";

    cin >> cm2;

    validare_comanda_general(cm2);

    meniu_modificare(cm2, ls, lp, ld);

    break;

case 4:

    ui.afisare_interf_sterge();

    cout << "Efectuati comanda: ";

    cin >> cm2;

    validare_comanda_general(cm2);

    meniu_stergere(cm2, ls, lp, ld);

    break;

```

```
case 5:  
    cout << "Program inchis cu succes.\n";  
    break;  
  
default:  
    cout << "Comanda invalida!\n";  
    break;  
  
}  
}  
  
_CrtDumpMemoryLeaks();  
  
return (0);  
}
```