Proiecte OOP (III C + CE) 2002

1. Rețele Petri (3 studenți)

Să se scrie o aplicație care permite următoarele operații pentru rețelele Petri binare:

- editor grafic pentru crearea/modificarea unei rețele;
- stocarea unei rețele Petri într-un fişier text conform unei sintaxe specificate;
- simularea execuției unei rețele Petri;
- afişarea grafică a evoluției execuției rețelei.

Editorul grafic permite:

- încărcarea unei rețele dintr-un fişier text;
- crearea rețelei cu ajutorul unei palete de obiecte;
- modificarea structurii rețelei;
- salvarea rețelei intr-un fişier text;
- lansarea simulatorului;
- afişarea evoluției atomilor în timpul procesului de simulare.

2. Sisteme liniare (2 studenți)

Să se scrie o aplicație ce permite următoarele operații cu sisteme liniare:

- citirea datelor de intrare;
- rezolvarea sistemelor prin una din metodele: Gauss (cel puțin două metode de pivotare), factorizare, metode iterative (Jacobi, Gauss-Seidel);
- afişarea soluțiilor.

Coeficienții ecuațiilor pot fi reali sau complecşi.

Cerințe:

- Utilizarea mecanismului *template* pentru unele dintre clase, astfel încât să permită sisteme cu ambele tipuri de coeficienți;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

3. Sisteme neliniare (2 studenți)

Să se scrie o aplicație ce permite următoarele operații cu sisteme neliniare:

- citirea datelor de intrare;
- rezolvarea sistemelor prin cel puțin două metode;
- afişarea soluțiilor.

Coeficienții ecuațiilor pot fi reali sau complecşi.

Cerințe:

- Utilizarea mecanismului *template* pentru unele dintre clase, astfel încât să permită sisteme cu ambele tipuri de coeficienți;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

4. Arbori oarecare (2 studenți)

Să se scrie o aplicație pentru operațiile cu arbori oarecare. Un nod dintr-un arbore indică spre următorul nod frate, precum şi spre lista nodurilor fii.

Operații:

- Creare arbore;
- Afişarea grafică a arborelui;
- Căutarea unui nod;
- Modificarea informațiilor dintr-un nod;
- Parcurgerea arborelui pe nivele;

Pentru operațiile de căutare şi parcurgere se va utiliza o structură de tip *coadă*.

Cerințe:

- Utilizarea mecanismului *template* pentru unele dintre clasele folosite, astfel încât informația din noduri să poată fi de tip: caracter, numeric sau şir de caractere;
- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare şi restaurare a datelor pentru anumite exemple;

- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

5. Arbori B (2 studenți)

Să se scrie o aplicație pentru operațiile cu arbori B.

Operații:

- Creare arbore;
- Afişarea grafică a arborelui;
- Căutarea unui nod;
- Modificarea informațiilor dintr-un nod;
- Inserare nod;
- Ştergere nod.

Cerințe:

- Utilizarea mecanismului *template* pentru unele dintre clasele folosite, astfel încât informația din noduri să poată fi de tip: caracter, numeric sau şir de caractere;
- Posibilitatea de stocare şi restaurare a datelor pentru anumite exemple;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

6. Arbori AVL (2 studenți)

Să se scrie o aplicație pentru operațiile cu arbori AVL.

Operații:

- Creare arbore;
- Afişarea grafică a arborelui;
- Căutarea unui nod;
- Modificarea informațiilor dintr-un nod;
- Inserare nod;
- Ştergere nod.

Cerințe:

- Utilizarea mecanismului *template* pentru unele dintre clasele folosite, astfel încât informația din noduri să poată fi de tip: caracter, numeric sau şir de caractere;
- Posibilitatea de stocare şi restaurare a datelor pentru anumite exemple;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

7. Evaluarea expresiilor algebrice (2 studenți)

Să se scrie o aplicație pentru evaluarea expresiilor algebrice cu numere reale sau complexe.

O expresie poate conține operanzi, operatori şi perechi de paranteze. Operanzii unei expresii pot fi constante sau nume de variabile. Operatorii pot fi:

- pentru expresiile reale: operatori aritmetici elementari (+, -, *, /) sau apeluri de funcții matematice elementare: sin, cos, arctg, sqrt, pow, log (se consideră funcțiile matematice drept operatori unari sau binari).
- Pentru expresiile complexe: operatori aritmetici elementari (+, -, *, /) sau apeluri de funcții complexe elementare: sqrt, pow, cabs (se consideră funcțiile complexe drept operatori unari).

Memorarea unei expresii algebrice se face printr-un arbore binar, la care nodurile frunză reprezintă operanzii, iar nodurile interne operatorii. Evaluarea expresiei se va face prin parcurgerea arborelui asociat.

Operații:

- Citirea formei textuale a expresiei;
- Citirea valorilor curente ale variabilelor care apar în expresie;
- Evaluarea expresiei;
- Modificarea valorilor unor variabile;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

Cerințe:

- Utilizarea mecanismului *template* pentru unele dintre clasele folosite;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice;
- Tratarea excepțiilor.

8. Gestiunea unei biblioteci (2 studenți)

Informațiile aferente cărților unei biblioteci sunt:

- nume carte;
- nume autor;
- cod ISBN;
- domeniu;
- data intrării în bibliotecă;
- număr cod (pot fi mai multe cărți de acelaşi fel, dar cu numere de cod diferite);
- dacă este împrumutată la data curentă;
- data ultimului împrumut;
- numele clinetului care a împrumutat-o;

Operații:

- introducere date;
- actualizare cărți:
  - ştergere carte din bibliotecă;
  - adăugare carte nouă;
- operații de împrumut/restituire carte;
- afişare cărți împrumutate sortate după data împrumutului;
- afişarea clienților care au împrumutată o anumită carte;
- afişare cărți neîmprumutate sortate după nume autor;
- afişarea cărților împrumutate de un client;
- afişarea cărților unui anumit domeniu sortate după nume autor.

Cerințe:

- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare şi restaurare a datelor;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

9. Gestiunea unei farmacii (2 studenți)

Evidența medicamentelor într-o farmacie (nume farmacie, adresă, listă de medicamente).
Pentru fiecare medicament se consideră următoarele informații:

- nume medicament;
- cod medicament (unic pentru un medicament dintr-o farmacie);
- număr bucăți;
- nume furnizor;
- data expirării;
- numele grupei de medicamente din care face parte;

Operații:

- introducere date;
- actualizare medicamente:
  - intrare medicament nou în farmacie;
  - cumpărare medicament (nume, număr bucăți);
- afişare medicamente (sortate după un anumit criteriu);
- determinarea cantității unui anumit medicament;
- eliminare produse cu cantitatea zero;
- afişarea medicamantelor cu o cantitate mai mare decât una specificată;
- afişrea medicamentelor de la un anumit furnizor sortate după nume;
- sortare medicamente după nume;
- sortare medicamente după preț;
- eliminare produse cu timul expirat;

Cerințe:

- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare şi restaurare a datelor;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

10. Gestiunea unui depozit (2 studenți)

Evidența produselor într-un depozit (nume depozit, adresă, listă produse).

Pentru fiecare produs se consideră următoarele informații:

- nume produs;
- cod produs (unic pentru fiecare produs din depozit);
- număr bucăți din același produs;
- preț unitar;

Operații:

- introducere date;
- actualizare produse:
  - intrarea în depozit a unor produse;
  - ieșirea din depozit a unor produse.
- sortare după cantitate produse;
- sortare după nume produs;
- sortare produse după cod;
- determinarea cantității unui anumit produs;
- eliminare produse cu cantitatea zero;
- determinarea produsului cu cantitate maximă;
- afișare produse sortate după cantitate;
- afișare produse sortate după cod;
- afișare produse sortate după nume;
- afișarea produselor cu o cantitate mai mare decât una specificată;
- comasarea depozitului curent cu un alt depozit specificat;

Cerințe:

- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare și restaurare a datelor;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

11. Secretariatul unei facultăți (2 studenți)

Gestiunea unui an de studii (numărul anului, facultatea și secția de care aparține, o listă pt. fiecare grupă din an, precum și un tablou cu toți studenții anului).

Pentru fiecare student se consideră următoarele operații:

- nume student;
- grupă;
- note sesiunea din iarnă (necunoscute la creare);
- note sesiunea din vară (necunoscute la creare);
- note sesiunea din toamnă (necunoscute la creare);

Operații:

- introducere date;
- adăugarea notelor la sesiunile din iarnă, vară, toamnă;
- interclasare liste (se formează tabloul cu toți studenții anului);
- afișare studenți bursieri după fiecare sesiune ordonați după medie;
- sortare liste după nume;
- adăugare studenți transferați (lista rămâne ordonată);
- afișare studenți pe grupe după nume;
- afișare studenți bursieri după fiecare sesiune ordonați după medie;
- eliminare studenți nepromovați în toamnă;
- eliminare studenți transferați;
- determinare procente de studenți promovați și restanțieri pe grupe și an de studiu după sesiunea din iarnă, vară, toamnă;

Cerințe:

- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare și restaurare a datelor;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

12. Evidența locatarilor dint-o asociație de blocuri (2 studenți)

Pentru fiecare apartament se conisderă următoarele informații:

- nume proprietar;

- număr apartament;
- număr bloc;
- număr persoane;
- sumă întreținere pe luna curentă (nu se cunoaşte în momentul creării);
- sume restante (număr luni, valoare restantă);

Operații:

- introducere date;
- actualizare date:
  - actualizare număr persoane;
  - plata întreținerii;
- determinarea restanțelor pe blocuri şi total asociație;
- afişare locatari ordonați după blocuri şi număr apartament;
- afişare chirii pe luna curentă pe blocuri şi apartamente;
- afişare locatari cu plata întreținerii restantă ordonați după număr luni restanțe, blocuri şi nume;
- afişare locatari nerestanți cu plata întreținerii ordonați după nume şi sumă întreținere;
- afişare locatari cu plata întreținerii restantă ordonați după total sumă restanțe, blocuri şi nume;
- determinarea sumelor restante în luna curentă pe blocuri şi total asociație;
- afişare chirii pe luna curentă pe blocuri şi apartamente;
- determinarea sumelor pentru întreținere pe blocuri şi total asociație;

Cerințe:

- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare şi restaurare a datelor;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

13. Carte de telefon (2 studenți)

Despre abonații telefonici se cunosc informațiile:

- nume abonat;
- număr telefon;
- adresă (stradă, număr casă sau bloc);

Operații:

- creare carte de telefon cu abonații sortați după nume;
- actualizare carte de telefon;
- listare abonați;
- afişare abonați care locuiesc pe o anumită stradă;
- determinarea telefonului unui abonat specificat prin nume;
- determinarea numelui unui abonat specificat prin număr telefon;

Cerințe:

- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare şi restaurare a datelor;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

14. Operații cu plata personalului într-o companie (2 studenți)

Pentru fiecare angajat se păstrează următoarele informații:

- nume;
- număr de cod;
- rata de plată pe oră;
- numărul de dependenți;
- codul de asigurare a sănătății;
- numărul de ore lucrate într-o lună;

Salariul brut se calculează din numărul de ore lucrate şi rata de plată pe oră; peste 40 de ore lucrate, rata de plată se multiplică cu 1.5. Diversele taxe care se rețin se calculează astfel:

- impozitul:

  f=t*(0.14+2.3*0.0001*t),

  unde t este venitul impozabil:

  t=g-14*e-11,

  unde g este salariul brut, iar e numărul de dependenți;

- taxa de securitate socială, are este 1.7% din salariul brut;

Salariul net se obține scăzând din salariul brut valorile tuturor impozitelor;
Opearații:

- introducere date;
- sortare salariați după nume;
- afişare salariați
- determinare salariu brut;
- afişare salariați cu salariu brut mai mare decât o sumă dată;
- determinare taxe şi impozite;
- afişare salariați ordonați după salariu brut;
- determinarea salariului brut maxim;
- determinare salariu net;
- determinarea salariului net minim;

Cerințe:

- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare şi restaurare a datelor;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

15. Birou de turism (2 studenți)

Informațiile aferente stațiunilor care oferă concedii sunt următoarele:

- nume stațiune;
- listă de perioade de sejur, specificate prin data de început şi sfârşit a sejurului;

Informațiile despre clienți sunt următoarele:

- nume şi prenume;
- număr şi serie buletin;
- adresă;
- număr de telefon;
- perioada de sejur dorită;
- lista stațiunilor dorite (în ordinea descrescătoare a preferințelor).

Operații:

- introducere date;
- afişare stațiuni sortate după nume;
- actualizare date:
  - adăugare a unei noi stațiuni cu perioadele aferente la data curentă;
  - adăugare a unei noi perioade la o stațiune;
- ştergerea stațiunilor care nu mai au perioade de sejur;
- afişare stațiuni care au bilete într-o anumită perioadă sortate după nume (listă);
- afişare perioade sejur pentru o anumită stațiune sortate după ziua de începere a perioadelor;
- afişare clienți care doresc aceeaşi perioadă într-o stațiune;
- afişare clienți care doresc o stațiune;
- determinarea numărului de locuri într-o anumită perioadă la o stațiune;

16. Operații bancare (2 studenți)

Evidența clienților şi a conturilor bancare într-o bancă.
Pentru fiecare client se consideră următoarele informații:

- nume nume şi prenume;
- număr şi serie carte de identitate;
- cod numeric personal;
- codul atribuit de bancă;
- lista conturilor deschise;

Pentru fiecare cont se cunoaşte

- codul clientului proprietar;
- depozitul curent în lei;
- depozitul curent în valută.

Operații:

- introducere date;
- depunere lei într-un cont al unui client;

- depunere valută într-un cont al unui client;
- retragere lei dintr-un cont al unui client;
- retragere valută dintr-un cont al unui client;
- deschidere cont pentru un client;
- ştergere cont al unui client;
- afişarea conturilor unui client;
- afişrea sumei aflată într-un cont;
- afişrea sumei totale aflate în conturile unui client;
- afişare clienţi;
- afişare conturi;
- balanța zilei curente.

Cerințe:

- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare şi restaurare a datelor;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

17. Automat bancar (2 studenți)

Să se simuleze un automat bancar. Un automat bancar poate accepta tranzacții bancare pentru o anumită băncă, posesoare a automatului. Un card este emis de o anumită bancă şi este asociat unui client al băncii şi unui anumit cont al clientului deschis în banca respectivă. El memorează suma de bani curentă din cont.
Interfața trebuie să permită principalele operații efectuate în timpul unei sesiuni de lucru:

- Deschiderea unei sesiuni de lucru;
- Închiderea sesiunii de lucru;
- Efectuarea principalelor tranzacții:
  - extragerea unei sume de bani dintr-un cont;
  - afişarea soldului contului curent;
  - transferul unei sume de bani din contul curent într-un alt cont al clientului de la aceeaşi bancă.

Deschiderea unei sesiuni de lucru începe prin introducerea unui card în automat şi verificarea validității informațiilor de pe card.

Cerințe:

- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare şi restaurare a datelor;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

18. Aplicație grafică (2 studenți)

Să se realizeze o aplicație grafică ce permite crearea şi editarea unor figuri geometrice plane:

- segmente de dreaptă;
- linii pologonale deschise;
- linii pologonale închise;
- cercuri;
- triunghiuri;
- pătrate;

Figurile geometrice pot avea diferite culori şi pot fi create (cu o dimensiune implicită), pot fi deplasate şi li se pot modifica dimensiunile. De asemenea, figurile dintr-un desen se pot salva într-un fişier, iar un desen poate fi restaurat dintr-un fişier. Formatul datelor din fişierele de stocare, precum şi formatul fişierelor (text sau binar) este ales de către studenți.

19. Cabinet medical pentru un doctor de familie (2 studenți)

Aplicația trebuie să țină evidența pacienților unui doctor de familie.
Pentru fiecare pacient se consideră următoarele informații:

- nume şi prenume;
- adresă;
- cod numeric personal;
- cod carnetului de sănătate;
- lista cronologică a consultațiilor pacientului; pentru fiecare consultație se memorează:
  - diagnosticul determinat;
  - rețeta prescrisă (dacă este cazul);

- trimiterea pacientului la un medic specialist (dacă este cazul);

Operații:

- venirea unui nou pacient în circumscripția doctorului de familie;
- plecarea unui pacient din circumscripția doctorului de familie;
- adăugarea informațiilor pentru o consultație;
- afişarea tuturor pacienților;
- afişarea tuturor consultațiilor unui pacient;
- afişarea consultațiilor unui pacient după o anumită dată;
- vizualizarea rețetei unei consultații;

Cerințe:

- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare şi restaurare a datelor;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.

20. Concurs de admitere (2 studenți)

Concurs de admitere la facultate (se specifică numele facultății şi data de susținere a concursului).
Pentru fiecare concurent se cunosc următoarele informații de intrare:

- nume şi prenume;
- data naşterii;
- numărul şi seria buletinului de identitate;
- pentru fiecare din cele două probe, trei note ale celor trei corectori;

Operații:

- introducere date;
- sortare concurenți după nume;
- calcul notă la fiecare probă;
- calcul medie;
- sortare concurenți după medie, iar pentru medii egale după nume;
- separarea în 2 liste (admişi, respinşi);
- determinarea numărului de studenți admişi şi respinşi;
- afişarea studenților admişi şi respinşi în ordinea descrescătaore a mediilor;
- rezolvarea contestațiilor:
  - modificarea notelor la candidații cărora li s-au rezolvat contestațiile;
  - inserarea (după contestații) a studenților cu o medie mai mare sau egală cu cea a ultimului admis în lista cu studenții admişi pe poziția corespunzătoare;
  - ştergerea din lista studenților respinşi a celor admişi după contestație;
  - schimbarea poziției fiecărui student admis, căruia după contestație i s-a modificat media;

Cerințe:

- Utilizarea claselor *iterator* pentru parcurgerea listelor;
- Posibilitatea de stocare şi restaurare a datelor;
- Tratarea excepțiilor;
- Dialogul cu utilizatorul prin intermediul unei interfețe grafice.