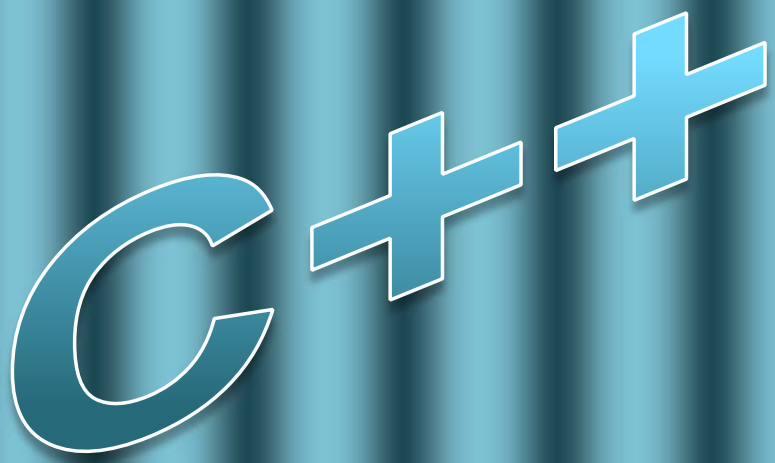
INFORMATICĂ

UNIVERSITATEA DE STAT TIRASPOL

COLEGIUL FINANCIAR - BANCAR

Silviu GÎNCU

Metodologia rezolvării problemelor de informatică în stilul orientat pe obiecte



Aprobat de Senatul Universităţii de Stat Tiraspol

Recenzenţi: *Andrei **Braicov***, doctor, conferenţiar universitar, UST

*Alexandru **Lazari***, doctor, lector superior, USM

DESCRIEREA CIP A CAMEREI NAŢIONALE A CĂRŢII

Gîncu, Silviu.

Metodologia rezolvării problemelor de informatică în stilul orientat pe obiecte / Silviu Gîncu; Univ. de Stat Tiraspol, Catedra Informatică şi Tehnologii Informaţionale. – Ch. : Univ. de Stat Tiraspol, 2012. – 112 p. – (Informatica).

60 ex.

ISBN 978-9975-76-085-0.

37.016.046:004

G 57

ISBN 978-9975-76-085-0.

Prefaţă

Pentru a rezolva probleme
trebuie să rezolvi probleme

Stilul de programare orientat pe obiecte (POO) este utilizat pentru rezolvarea problemelor de informatică cu un grad sporit de dificultate. Conform acestui stil programul constă dintr-o colecţie de obiecte, unităţi individuale de cod care interacţionează între ele.

Elaborarea unui program în stilul POO presupune a organiza codul acestuia în unităţi de program care să opereze cu obiecte. În acest sens elaborarea programului constă în prelucrarea obiectelor existente sau crearea lor în dependenţă de specificul problemei.

Prezenta lucrare ştiinţifico-didactică cuprinde o gamă variată de programe rezolvate şi o serie de probleme pentru rezolvare independentă. Este o continuare a ghidului de iniţiere „Borland C++ Builder" (lucrarea 2 din referinţele bibliografice).

Este adresată studenţilor specialităţii informatică, profesorilor de informatică cît şi tuturor celor care sunt interesaţi de tehnologia POO.

Problemele rezolvate au ca scop prezentarea principalelor concepte care stau la baza POO. În baza acestora este prezentată metodologia de creare a unei clase; de definirea datelor şi prelucrarea metodelor; de iniţializare a datelor obiectului prin intermediul constructorului; de creare a relaţiilor dintre clase (moştenire) şi a relaţiilor dintre obiecte (agregare). Programele sunt rezolvate în baza limbajului de programare C++, limbaj ce se află la temelia mai multor medii de programare vizuală (C++ Builder, Visual C++, etc.).

Un salt important în dezvoltarea tehnologiilor informaţionale s-a produs odată cu apariţia mediilor de programare vizuală orientate pe obiecte. Pentru formarea deprinderilor de realizare a aplicaţiilor Windows orientate pe obiecte sunt prezentate o serie de aplicaţii care ilustrează modalitatea de gestionare a componentelor, de utilizare a elementelor de grafică, de creare a noi tipuri de date şi de utilizare a lor într-un mediu de programare vizuală. Drept mediu de programare este utilizată platforma Borland C++ Builder.

Pentru dezvoltarea abilităţilor de programare, recomand cititorilor rezolvarea a cît mai multe probleme.

Autorul

Sugestii metodologice

1. Modelul orientat pe obiecte

Conform Grady Booch „tehnologia orientată pe obiecte a fost concepută în baza principiilor de abstractizare, incapsulare, modularitate, ierarhizare, tipizare, paralelilism şi persistenţă. Ceea ce este important constă în faptul că aceste principii se regăsesc în modelul orientat pe obiecte.”[4, p.25].

1. *Abstractizare* - procesul care captează caracteristicile esenţiale ale unui obiect, caracteristici ce diferenţiază acest obiect de toate celelalte tipuri de obiecte. Prin abstractizare se rezolvă problemele cu un grad sporit de complexitate. Abstractizarea se manifestă prin determinarea similarităţii dintre anumite obiecte, situaţii sau procese din lumea reală şi în luarea deciziilor, ignorînd pentru moment deosebirile dintre obiecte;

2. *Incapsulare* - procesul mental executat pe o abstractizare în care elementele structurale ale abstractizării sunt izolate de cele ce constituie comportamentul său. Serveşte la separarea interfeţei vizibile a unei abstractizări de implementarea sa;

3. *Modularitate* - procesul de divizare a unui sistem complex în părţi (module) manevrabile. Modularitatea este caracteristica de divizare a programului în mai multe unităţi care sunt compilate separat, dar simultan şi comunică între ele;

4. *Ierarhizare* - procesul de clasificare pe nivele de abstractizare;

5. *Tipizare* – reprezintă un mod de protejare a obiectelor. Prin intermediul tipizării se face distincţie dintre obiectele diferitor clase, astfel obiectele nu pot fi schimbate sau, în unele cazuri, această schimbare este limitată;

6. *Concurenţa (paralelism)* – este o proprietate, prin intermediul căreia se permite ca mai multe obiecte diferite să fie în execuţie în acelaşi timp. Sistemele care implică concurenţa sunt sisteme cu procesoare multiple, în care mai multe procese sau thread-uri (fire de execuţie) pot fi executate concurent pe procesoare diferite.

7. *Persistenţa* – este proprietatea obiectelor care implică existenţa acestora şi după încetarea procesului care le-a creat. Starea obiectului şi codul corespunzător metodelor sunt memorate în baza de date. Tipurile obiectelor pot fi declarate persistente prin folosirea cuvîntului cheie persistent la momentul declarării, variabila fiind şi ea constrînsă la un tip persistent.

Esenţiale pentru limbajele de programare orientate pe obiect sunt primele patru caracteristici. Un limbaj de programare este orientat pe obiecte dacă susţine primele patru caracteristici.

2. Caracteristici ale programării orientate pe obiecte

Din perspectiva programării, rezolvarea unei probleme se poate face pe 3 direcţii:

a) Rezolvarea orientată pe algoritm;
b) Rezolvarea orientată pe date;
c) Rezolvarea orientată pe obiect, combină tendinţele primelor două abordări.

Programarea *orientată pe obiecte* POO presupune unirea datelor cu subprogramele care prelucrează aceste date într-un tot întreg, numit obiect. Acest stil de scriere a codului, numit tehnologie, oferă posibilităţi de modelare a obiectelor, a proprietăţilor şi a relaţiilor dintre ele, dar şi posibilitatea de a descompune o problemă în componentele sale (soft mai mentenabil, adaptabil, reciclabil).

Acest mod de rezolvare se bazează pe crearea de obiecte şi pe interacţiunea dintre ele. Cele mai importante caracteristici ale tehnologiei orientate pe obiecte sunt:

Obiectul – asociază datele împreună cu operaţiile necesare prelucrării. Datele sunt informaţii de structură descrise de o mulţime de atribute ale obiectului, iar operaţiile (metodele) acţionează asupra atributelor obiectului şi eventual, asupra altor obiecte;

Clasa – reuneşte o colecţie de obiecte care partajează aceeaşi listă de atribute informaţionale şi comportamentale;

Incapsularea – principiul care se bazează pe combinarea datelor cu operaţiile de gestionare a obiectului şi proprietatea obiectelor de a ascunde datele şi operaţiile proprii faţă de alte obiecte;

Instanţierea –operaţia prin care se creează un obiect şi apoi se iniţializează cu date specifice;

Moştenirea – principiul prin care putem reutiliza şi extinde clasele existente. Acest mecanism permite unei noi clase să beneficieze de atributele şi metodele definite într-o clasă deja existentă;

Agregarea – operaţiile cu ajutorul cărora clasele sunt descompuse în unităţi mai mici, care la fel sunt clase. O clasă agregat este o clasă ale cărei obiecte conţin alte obiecte.

Polimorfismul – posibilitatea de a putea aplica în moduri diferite o aceeaşi operaţie mai multor clase. Prin operaţie înţelegem orice metodă sau operator. Polimorfismul reprezintă abilitatea unor obiecte similare de a răspunde la acelaşi mesaj în moduri diferite.

3. Limbaje de programare orientate pe obiecte

Un limbaj de programare reprezintă un set de specificaţii necesare reprezentării algoritmilor. Dezvoltarea limbajelor de programare a cunoscut în timp mai multe generaţii.

Prima generaţie a limbajelor de programare o constituie perioada 1954-1958. Limbajele din această generaţie au fost concepute pentru a simplifica complexitatea formulelor matematice. Sunt caracteristice limbajele: FORTRAN I, ALGOL-58, Flowmatic şi IPL V.

A doua generaţie datează anilor 1959-1961 şi se caracterizează prin concentrarea atenţiei asupra abstracţiunilor algoritmice. În acea perioadă calculatoarele au devenit mai puternice, ele fiind utilizate în soluţionarea problemelor de ordin economic, de evidenţă a personalului, etc. Sunt utilizate limbajele: FORTRAN II, ALGOL-60, COBOL, Lisp.

Odată cu apariţia tranzistoarelor, costul unui calculator a scăzut considerabil, astfel producerea lor a crescut esenţial. Creşterea productivităţii acestora a dus la apariţia unei noi generaţii a limbajelor de programare în perioada 1962-1970. Au apărut limbaje precum: PL/I, ALGOL-68, Pascal, Simula. Aceste limbaje se caracterizează prin faptul că oferă un alt nivel de abstractizare a datelor, oferind programatorului posibilitatea de a crea noi tipuri de date.

Perioada 1970-1980 se caracterizează prin apariţia mai multor limbaje de programare, care nu au făcut faţă cerinţelor pieţii. Problema abstractizării obiectelor a fost caracteristică perioadei 1970-1980, astfel au apărut limbajul Simula, iar în baza acestuia şi altele precum Smalltalk, Object Pascal, C++, CLOS, Ada şi Eiffel. Aceste limbaje au fost numite orientate pe obiecte.

Lansarea sistemului de operare Windows a impus firmele specializate să dezvolte produse noi, destinate implementării pe calculator a unor programe concepute în baza unui limbaj de programare. Astfel au apărut mai multe medii de programare vizuală, destinate dezvoltării rapide a aplicaţiilor Windows orientate pe obiecte. Mediile de programare conţin o gamă largă de biblioteci, care sunt compuse din clase. Spre exemplu, mediul de programare C++ Builder conţine biblioteca VCL (Visual Componente Library) formată din clase, funcţii şi variabile necesare pentru elaborarea de aplicaţii. O clasă, în mediul de programare vizuală, este „completată” cu două elemente noi:

Proprietăţi – reprezintă starea unui obiect, din punct de vedere sintactic datele unui obiect sunt transformate în proprietăţi, care au ataşat în faţă cuvîntul cheie **__property**.

Evenimente – reprezintă o legătură dintre o acţiune în sistem (apăsarea unei taste, deplasarea maus-ului, etc.) şi o secvenţă de cod care răspunde la acţiunea efectuată.

Instanţele unor astfel de clase sunt numite **componente**. „O componentă este definită ca fiind o clasă ce derivă direct sau indirect din clasa *TComponent*.” [5, p. 17]

Unitatea de bază a unei aplicaţii Windows o constituie componentele. Acestea pot fi grupate în:

Componente container – pot conţine pe suprafaţa lor alte componente. De exemplu: Form, GroupBox, etc.;

Componente de acţiune – destinate prelucrării unei acţiuni (efectuarea unui

click, etc.). De exemplu: Button şi alte derivate ale acesteia;

Componente valoare – prin intermediul cărora este prezentată o informaţie. De exemplu: Label, Edit, etc.;

Componente de selecţie (CheckBox, RadioButton, etc);

Componente de listare (ListBox, MonthCalendar, etc.);

Componente pentru prelucrarea grafică (Obiectul Canvas), etc.

Prin intermediul componentelor un mediu de programare oferă posibilitatea utilizatorului de a dezvolta aplicaţii Windows orientate pe obiecte.

4. Tendinţe

În prezent tehnologiile informaţionale se află în continuă creştere. Astfel, au apărut noi platforme de dezvoltare. Concomitent cu acestea s-au dezvoltat şi mediile de programare vizuală orientate pe obiecte. Ele oferă posibilitatea de a elabora diverse produse software. Prin intermediul lor se pot realizează:

a) aplicaţii Windows orientate pe obiecte;
b) aplicaţii Windows pentru gestiunea bazelor de date;
c) aplicaţii Windows pentru lucru în reţea;
d) aplicaţii pentru gestiunea paginilor web, etc.

Elaborare unei aplicaţii Windows orientată pe obiecte presupune:

1. A gestiona componentele oferite de către mediul de programare utilizat. Un mediu de programare vizuală orientat pe obiecte conţine o gamă largă de componente destinate procesării datelor. Pentru gestiunea lor, sunt necesare cunoştinţe în ceea ce priveşte structura obiectelor. Aceasta presupune a studia proprietăţile, metodele şi evenimentele caracteristice unei componente şi modalitate de modificare a lor în dependenţă de specificul problemei.

2. A elabora componente noi. În cazul în care mediul de programare nu oferă suficiente componente, apare tendinţa de elaborare a lor.

Observăm necesitatea studierii programării orientate pe obiecte în ambele cazuri.

5. Dificultăţi în însuşirea programării orientate pe obiecte

În prezent conform literaturii de specialitatea din domeniu [3],[4],[10], etc., şi a planurilor de învăţământ ale universităţilor din Republica Moldova şi nu numai, tehnologia programării orientată pe obiecte este formată în baza:

a) unui limbaj de programare de uz general, cele mai frecvent utilizate fiind Object Pascal, C++, C#;
b) unui mediu de programare cu elemente de programare vizuală, cele mai frecvent utilizate fiind: Borland Delphi/Builder, Java, Visual C++, Visual Basic, etc.

În timp s-a observat că studenţii întîmpină mai multe dificultăţi în însuşirea deprinderilor de utilizare a tehnologiei orientate pe obiecte. De cele mai multe ori acestea se datorează:

1. Lipsei unui model, care să se expună asupra conţinuturilor şi asupra metodelor de predare/învăţare.

2. Insuficienţei de probleme rezolvate, prin intermediul cărora să se prezinte modalitatea de elaborare a unui program în stilul de programare orientat pe obiecte.

3. Gradului sporit de dificultate al problemelor propuse spre rezolvare în cadrul orelor de laborator, etc.

6. Reflecţii privind conţinutul şi organizarea procesului de instruire

Pentru înlăturarea dificultăţilor întîlnite în procesul de studiu al POO se recomandă:

1. A utiliza modelul moderat de formare a competenţelor, elaborat de către un grup de cercetători din Belgia (X. Rogiers, J.-M. De Ketele, F.-M. Gérard). Conform modelului formarea competenţei se realizează în trei etape:

a) *Structurare* a resurselor. Această etapă presupune studierea materialului teoretic, concepte, noţiuni, definiţii, etc.;
b) *Integrare* a resurselor. Resursele teoretice sunt integrate prin prisma exemplelor. Fiecare dintre exemple avînd drept scop final prezentarea unei sau a mai multor caracteristici ale POO.
c) *Transfer* sau adaptarea la noi situaţii. În cadrul acestei etape sunt propuse probleme care diferă, dar nu considerabil de problemele din cadrul etapei a doua. Astfel dacă problema este rezolvată atunci transferul este realizat.

2. De a realiza un studiu de caz asupra unui proiect. Astfel vor fi prezentate avantajele oferite de către POO comparativ cu alte stiluri de programare.

3. A propune spre rezolvare la orele de laborator un set de probleme astfel încît:

a) Gradul de dificultate al problemelor să nu fie foarte ridicat;
b) Problemele propuse spre rezolvare să cuprindă majoritatea conceptelor POO studiate;
c) Problemele propuse să ofere continuitate în procesul de studiu;
d) Să realizeze un proiect care să cuprindă în structura sa mai multe module.

Capitolul I Limbajul C++

C++ (pronunțat în română „C plus plus” și în engleză „si plas plas”) este un limbaj de programare orientat pe obiecte. A fost creat de către Bjarne Stroustrup, lansarea comercială datînd anului 1985. În 1990 devine unul dintre cele mai populare limbaje de programare comerciale.

1. Crearea de obiecte

Problemă exemplu. Creaţi obiectul *dreptunghi*

date: *lungimile laturilor*;

metode: *citire, afişare, iniţializare, determinare a perimetrului şi ariei*.

Se vor introduce datele despre două dreptunghiuri şi se vor afişa datele despre dreptunghiul cu suprafaţa maximă şi dreptunghiul cu perimetrul minim.

Rezolvare :

```
#include<iostream.h>
class dreptunghi{
 double lung,lat;
 public:
 void citire();//pentru citirea datelor
 void afisare();//pentru afisare
 double arie();
 double perimetru();
 void init(double a, double b){lung=a;lat=b;}
};
void dreptunghi::citire(){
 cout<<"Dati datele despre dreptunghi"<<endl;
 cout<<"Lungimea="<<endl;cin>>lung;
 cout<<"Latimea"<<endl;cin>>lat;
}
double dreptunghi::arie(){return lung*lat;}
double dreptunghi::perimetru(){return 2*(lung+lat);}
void dreptunghi::afisare(){
 cout<<"Lungimea="<<lung<<endl;
 cout<<"Latimea="<<lat<<endl;
 cout<<"Aria="<<arie()<<endl;
 cout<<"Perimetru="<<perimetru()<<endl;
}
main(){
 dreptunghi a,b;
 a.citire();b.init(4.5,8);
 a.afisare();b.afisare();
 cout<<"Dreptunghiul cu suprafata maxima"<<endl;
 if(a.arie()>b.arie())a.afisare();
 else b.afisare();
 cout<<"Dreptunghiul cu perimetru minim"<<endl;
```

```
 if(a.perimetru()<b.perimetru())a.afisare();
 else b.afisare();
}
```

1. Creaţi obiectul *cerc*

date: *lungimea razei*;

metode: *citire, afişare, iniţializare, determinarea lungimii discului, a suprafeţei şi a diametrului*.

Se vor introduce datele despre două cercuri şi se vor afişa datele despre cercul cu suprafaţa maximă şi cercul cu lungimea discului minimală.

2. Creaţi obiectul *trapez_isoscel*

date: *baza mare, baza mică, înălţimea*;

metode: *citire, afişare, iniţializare, determinare a perimetrului şi ariei*.

Se vor introduce datele despre două trapeze şi se va afişa trapezul cu suprafaţa maximă şi trapezul cu perimetrul minim.

3. Creaţi obiectul *paralelogram*

date: *lungimile laturilor şi înălţimea*;

metode: *citire, afişare, iniţializare, determinare a perimetrului şi ariei*.

Se vor introduce datele despre două paralelograme şi se vor afişa datele despre paralelogramul cu suprafaţa maximă şi paralelogramul cu perimetrul minim.

4. Creaţi obiectul *triunghi*,

date: *lungimile laturilor*;

metode: *citire, afişare, iniţializare, determinare a perimetrului, a suprafeţei şi afişare a tipului triunghiului* (ascuţit, dreptunghic sau obtuzunghi).

Se vor introduce datele despre două triunghiuri şi se vor afişa datele despre triunghiul cu suprafaţa maximă şi triunghiul cu perimetrul minim. Se va implementa o metodă pentru verificarea corectitudinii introducerii datelor de intrare. În caz că lungimile introduse nu pot forma un triunghi, atunci lungimile laturilor vor fi egale cu 0, iar metoda tip va afişa mesajul: *Nu exista aşa triunghi*.

5. Creaţi obiectul *paralelipiped_dreptunghic*

date: *lungimile bazelor şi înălţimea*;

metode: *citire, afişare, iniţializare, determinare a suprafeţei laterale, suprafeţei totale, şi a volumului*.

Se vor introduce datele despre două paralelipipede dreptunghice şi se vor afişa datele despre paralelipipedul cu volumul maxim şi paralelipipedul cu suprafaţa totală minimă.

6. Creaţi obiectul *ecuaţie1* - va reprezenta o ecuaţie de gradul 1 $ax+b=0$.

date: *a, b*;

metode: *citire, afişare, iniţializare, soluţie*.

Se vor introduce datele despre două ecuaţii şi se vor afişa ecuaţiile introduse cît şi soluţiile acestora.

7. Creaţi obiectul *ecuaţie2* - va reprezenta o ecuaţie pătratică de ordinul doi $ax^2+bx+c=0$.

date: *a, b, c*;

metode: *citire, afişare, discriminant, soluţie*.

Se vor introduce datele despre două ecuaţii şi se vor afişa ecuaţiile introduse, discriminantul fiecărei ecuaţii cît şi soluţiile acestora.

8. Creaţi obiectul *persoana*,

date: *nume, anul naşterii, genul, înălţimea*;

metode: *citire, afişare, virsta, greutate* (se va determina conform formulelor:

Gmasculin=50+0.75(înălţime-150) + (vîrsta-20)/4 – pentru genul masculin*;

Gfeminin=Gmasculin–10 – pentru genul feminin.

De la tastatură se citesc datele despre *n* persoane. Să se afişeze:

e) Lista turor persoanelor;
f) Lista persoanelor cu o vîrstă mai mare decît 25 ani;
g) Lista persoanelor cu o greutate cuprinsă între 55 şi 80 kg.

9. Creaţi obiectul *student*

date: *nume, anul naşterii, notele la trei discipline*;

metode: *citire, afişare, virsta, media, restanţa* (va returna 1 doar dacă studentul are cel puţin o notă mai mică decît 5), *bursa* (va fi egală cu 500 lei dacă studentul nu are restanţă şi media este mai mare decît 7.5).

De la tastatură se citesc datele despre *n* studenţi. Să se afişeze:

a) Lista tuturor studenţilor;
b) Lista studenţilor restanţieri;
c) Lista studenţilor care au bursă;
d) Lista studenţilor cu o vîrstă mai mică decît 18 ani.

10. Creaţi obiectul *angajat*

date: *nume, funcţia, anul angajării, ore lucrate*, *plata pentru o oră*;

metode: *citire, afişare, stagiu* (diferenţa dintre anul curent şi anul angajării), *salariu calculat* (se va determina conform formulei: *ore lucrate* plata pentru o oră*), *spor* (va constitui 15% din salariul calculat, dacă stagiul este mai mare decît 5 ani şi

mai mic sau egal decît 8 ani şi 25%, dacă stagiu este mai mare decît 8 ani), *Salariu primit* (se va determina conform formulei: *salariu calculat+spor*).

De la tastatură se citesc datele despre *n* angajaţi. Să se afişeze:

a) Lista tuturor angajaţilor;
b) Lista angajaţilor cu un stagiu mai mare decît 10 ani;
c) Lista angajaţilor cu un salariu mai mare decît 1000;

2. Constructori

Clasă-pereche este numită o clasă cu două date, care de obicei sunt două numere *n1* şi *n2*. Este necesar de a elabora un program, prin intermediul căruia se va crea tipul de date *clasă-pereche*. În toate sarcinile trebuie să fie prezente:

a) Trei tipuri de constructori: fără parametri, cu parametri, de copiere;
b) Destructorul;
c) Metoda de citire;
d) Metoda de afisare;

Toate aceste condiţii necesită a fi respectate pentru următoarele cazuri.

Problemă exemplu Cîmpul n1 - număr real, Cîmpul n2 – număr real. Creaţi metoda suma() – va returna suma numerelor.

Rezolvare :

```
#include<iostream.h>
 class numar{
 double n1,n2;
 public:
 numar(){n1=n2=0;}
 numar(double a,double b){n1=a;n2=b;}
 numar(numar&);
 void citire();
 void afisare();
 double suma();
 ~numar(){cout<<"Obiectul a fost distrus !!!"<<endl;}
};
numar::numar(numar &c){// constructor de copiere
 n1=c.n1;n2=c.n2;
}
void numar::citire(){
 cout<<"Dati doua numere"<<endl;
 cin>>n1>>n2;
}
void numar::afisare(){
 cout<<"Valorile "<<n1<<" "<<n2<<" Suma="<<suma()<<endl;
}
double numar::suma(){return n1+n2;}
main(){
```

```
 numar a;//constructor fara parametri
 a.citire();a.afisare();
 numar b(3.2,8.5);//constructor cu parametri
 b.afisare();
 numar c(a);//constructor de copiere
 c.afisare();
 numar *p; //utilizarea pointerilor
 p=new numar;//apelarea constructorului fara parametri
 p->afisare();
 delete p;//distrugerea obiectului
 p=new numar(5,6);//apelarea constructorului cu parametri
 p->afisare();delete p;
 p=new numar(b);//apelarea constructorului de copiere
 p->afisare();delete p;
}
```

1. Cîmpul *n1* - un număr întreg, Cîmpul *n2* – număr real nenul. Creaţi metoda *power*() – va returna numărul *n2* la puterea *n1*.

2. Cîmpul *n1* - un număr întreg, Cîmpul *n2* – număr real. Creaţi metoda *intreg*() - care va returna partea întreagă a fracţiei *n1/n2*. Metoda trebuie să verifice inegalitatea numitorului cu zero.

3. Cîmpul *n1* - un număr real pozitiv (preţul bunurilor), Cîmpul *n2* – număr întreg pozitiv (numărul de bunuri). Creaţi metoda *cost*() – va returna numărul ce reprezintă valoarea totală a bunurilor.

4. Cîmpul *n1* - un număr întreg pozitiv (calorii la 100g), Cîmpul *n2* – număr real (greutatea produsului în kilograme). Creaţi metoda *calorii*() – va returna numărul de calorii ale produsului.

5. Cîmpul *n1* - un număr real (suma depozitul în lei), Cîmpul *n2* – număr real (procentul anual). Creaţi metoda *valoare*(*int*) – va returna suma de bani împreună cu calcularea dobînzii timp de 1, 2, 3, 4 ani.

6. Cîmpul *n1* - un număr întreg pozitiv (numărul de ore), Cîmpul *n2* – un număr întreg pozitiv (numărul de minute). Creaţi metoda *secunde*() – va returna numărul total al secundelor scrise în cele două cîmpuri.

7. Cîmpul *n1* - un număr real (coordonata x), Cîmpul *n2* – număr real (coordonata y). Creaţi metoda *distanta*() – va returna distanţa de la punctul (*0,0*) la punctul cu coordonatele (*n1, n2*).

8. Cîmpul *n1* - un număr real (cateta *a* a unui triunghi dreptunghic), Cîmpul *n2* număr real (cateta *b*). Creaţi metodele:

ipotenuza() – va returna lungimea ipotenuzei triunghiului dreptunghic;
arie() - va returna aria triunghiului;
perimetru() - va returna perimetrul triunghiului.

9. Cîmpul *n1* - un număr real (salariul pentru o zi), Cîmpul *n2* – număr întreg (numărul de zile lucrate timp de o lună). Creaţi metoda *suma*() – va returna salariul pentru un an al muncitorului.

10. Cîmpul *n1* - un număr întreg pozitiv (durata unui apel telefonic în secunde), Cîmpul *n2* – număr real (costul în lei al unui minut). Creaţi metoda *cost*() – va returna costul total al apelului.

3. Moştenire simplă

Problemă exemplu În calitate de clasă de *bază* se consideră clasa *binar*. În calitate de date vor fi două valori reale. Metodele vor fi:

a) constructorul cu şi fără parametri;
b) metodele de citire şi afisare.

Creaţi clasa *dreptunghi*, derivata clasei *binar*. Pentru clasa dreptunghi vor fi implementate metodele de determinare a suprafeţei şi a perimetrului.

De la tastatură se citesc datele despre *n* dreptunghiuri. Elaboraţi un program prin intermediul căruia la ecran se vor afişa datele despre toate dreptunghiurile, dreptunghiul cu suprafaţa maximă şi dreptunghiul cu perimetrul maxim.

Rezolvare :

```
#include<iostream.h>
class binar{
 protected:
 double a,b;
 public:
 binar(){a=b=0;}
 binar(int i, int j){a=i;b=j;}
 void citire();
 void afisare();
};
class dreptunghi : public binar{
 public:
 dreptunghi(){};
 dreptunghi(int i,int j):binar(i,j){};
 double arie();
 double perimetru();
};
void binar::citire(){cout<<"Dati 2 valori reale "; cin>>a>>b;}
void binar::afisare(){cout<<"a="<<a<<" B="<<b<<endl;}
double dreptunghi::arie(){return a*b;}
```

```
double dreptunghi::perimetru(){return 2*(a+b);}
main(){
 dreptunghi t[100];
 int am,pm,i,n;double ma=0,mp=0;
 cout<<"Dati numarul dreptunghiurilor ";cin>>n;
 for(i=0;i<n;i++){
  t[i].citire();
  if(ma<t[i].arie()){ma=t[i].arie();am=i;}
  if(mp<t[i].perimetru()){mp=t[i].perimetru();pm=i;}
 }
 cout<<"Lista dreptunghiurilor"<<endl;
 for(i=0;i<n;i++)t[i].afisare();
 cout<<"Perimetru maxim "<<endl;
 t[pm].afisare(); cout<<t[pm].perimetru()<<endl;
 cout<<"Aria maxima "<<endl;
 t[am].afisare(); cout<<t[am].arie()<<endl;
 dreptunghi a(8,10);a.afisare();
 cout<<"Aria "<<a.arie()<<endl;
 cout<<"Perimetru "<<a.perimetru()<<endl;
}
```

A. În calitate de clasă de *bază* se va crea clasa *triad*.
date: trei numere reale;
metode: *constructorul* cu/fără parametri, *citire* şi *afisare*.

1. Creaţi clasa *triunghi*, derivata clasei *triad*. Pentru clasa triunghi vor fi implementate metode pentru determinare a suprafeţei, a perimetrului şi o metodă prin intermediul căreia se va verifica dacă datele introduse pot fi lungimile laturilor unui triunghi.

De la tastatură se citesc datele despre *n* triunghiuri. Elaboraţi un program prin intermediul căruia, la ecran se vor afişa datele despre toate triunghiurile, triunghiul cu suprafaţă maximă şi triunghiul cu perimetru maxim.

2. Creaţi clasa *paralelogram*, derivata clasei *triad*. Pentru clasa paralelogram vor fi implementate metodele de determinare a suprafeţei şi a perimetrului.

De la tastatură se citesc datele despre *n* paralelograme. Elaboraţi un program prin intermediul căruia la ecran se vor afişa datele despre toate figurile, paralelogramul cu suprafaţă maximă, paralelogramul cu perimetru maxim.

3. Creaţi clasa *trapez_isoscel*, derivata clasei *triad*. Pentru clasa trapez isoscel vor fi implementate metodele de determinare a suprafeţei şi a perimetrului.

De la tastatură se citesc datele despre *n* trapeze isoscele. Elaboraţi un program prin intermediul căruia la ecran se vor afişa datele despre toate figurile, trapezul cu suprafaţă maximă, trapezul cu perimetru maxim.

4. Creaţi clasa *prismă* (baza este dreptunghi), derivata clasei *triad*. Pentru clasa prismă vor fi implementate metodele de determinare a suprafeţei totale, laterale şi de determinare a volumului.

De la tastatură se citesc datele despre *n* prisme. Elaboraţi un program prin intermediul căruia, la ecran se vor afişa datele despre toate prismele, prisma cu suprafaţă maximă, prisma cu volumul maxim.

5. Creaţi clasa *piramidă* (baza este dreptunghi), derivata clasei *triad*. Pentru clasa piramidă vor fi implementate metodele de determinare a suprafeţei totale, laterale şi de determinare a volumului.

De la tastatură se citesc datele despre *n* piramide. Elaboraţi un program prin intermediul căruia, la ecran se vor afişa datele despre toate piramidele, piramida cu suprafaţă maximă, piramida cu volumul maxim.

B. În calitate de clasă de *bază* se va crea clasa *persoana*.

date: *nume*, *prenume*, *anul*, *cnp* (codul personal de 13 cifre);

metode : *citire*, *afişare, virsta*.

6. Creaţi clasa *Salariat*, derivata clasei *persoana*. Pentru clasa salariat vor fi adăugate:

date: numărul de ore lucrate, plata pentru o oră, anul angajării;

metode: vor fi redefinite metodele de *citire* şi *afişare*, *salariu*. Salariul va fi determinat conform formulei : *nr_ore*plata*.

De la tastatură se citesc datele despre *n* salariaţi. Elaboraţi un program prin intermediul căruia la ecran se vor afişa datele despre toţi salariaţii, salariatul cel mai tînăr şi salariatul cu salariu maxim.

7. Creaţi clasa *Student*, derivata clasei *persoana*. Pentru clasa student vor fi adăugate:

date: media notelor curente, grupa;

metode: vor fi redefinite metodele de *citire* şi *afişare*, *bursa*. Bursa va fi determinată conform fomulei: *media*75 lei,* dacă media este mai mare decît 7.5, în caz contrar va fi 0.

De la tastatură se citesc datele despre *n* studenţi. Elaboraţi un program prin intermediul căruia la ecran se vor afişa datele despre toţi studenţii, studentul cel mai tînăr, studentul cu bursa maximă.

8. Creaţi clasa *Sofer*, derivata clasei *persoana*. Pentru clasa şofer vor fi adăugate:

date: distanţa parcursă (în km), plata pentru 1 km;

metode: vor fi redefinite metodele de *citire* şi *afişare*, *plata*. Prin intermediul metodei *Plata*, se va determina costul unei curse efectuate de către şofer după formula: *distanta*plata pentru 1 km*.

De la tastatură se citesc datele despre *n* şoferi. Elaboraţi un program prin intermediul căruia la ecran se vor afişa datele despre toţi şoferii, şoferul cel mai tînăr, şoferul cu plata maximă.

9. Creaţi clasa *Angajat_calificat*, derivata clasei *Salariat* (prob. 6). Pentru clasa *Angajat_calificat* vor fi adăugate:

date: gradul calificării (1 superior, 2 mediu, 3 inferior);

metode: vor fi redefinite metodele de *citire* şi *afişare*, *salariu*. Salariu va fi determinat conform formulei : *nr_ore*plata*(100%+calificare),*

$$calificare = \begin{cases} 50\%, pentru\ \ gradul\ \ 1 \\ 40\%, pentru\ \ gradul\ \ 2 \\ 30\%, pentru\ \ gradul\ \ 3 \end{cases}$$

De la tastatură se citesc datele despre *n* angajaţi calificaţi. Elaboraţi un program prin intermediul căruia la ecran se vor afişa datele despre toţi angajaţii, angajatul cel mai tînăr şi angajatul cu salariu maxim.

10. Creaţi clasa *Piramida*, derivata clasei *triunghi* (prob. 1).

date: *înălţimea* piramidei,

metode: *citire* şi *afişare*, *suprafaţa_totală* şi *volum*.

De la tastatură se citesc datele despre *n* piramide şi *m* triunghiuri. Elaboraţi un program prin intermediul căruia, la ecran se vor afişa datele despre toate piramidele (triunghiurile), piramida (triunghiul) cu suprafaţă maximă, piramida cu volumul maxim, triunghiul cu perimetrul minim.

4. Funcţii virtuale şi polimorfism

Problemă exemplu. [2, p. 180].

1. Se consideră drept bază clasa *numar* (va avea în calitate de date un număr real). În baza acestei clase se derivă clasele patrat şi triunghi_echilateral. Să se implementeze polimorfismul pentru metodele: *citire*, *afişare*, *suprafaţa* şi *perimetru*. De la tastatură se citeşte numărul de figuri. Elaboraţi un program care va permite :

a) citirea figurilor de la tastatură;
b) afişarea acestora;

c) afişarea figurilor sub formă de pătrat;
d) afişarea figurilor sub formă de triunghi;
e) afişarea figurii cu suprafaţă maximă;
f) afişarea figurii cu perimetru maxim;
g) afişarea perimetrului total al figurilor;
h) afişarea suprafeţei totale a figurilor;
i) sortarea figurilor în ordine descrescătoare suprafeţei.

2. Se consideră drept bază clasa *doi* (va avea în calitate de date două numere reale). În baza acestei clase se derivă clasele dreptunghi şi triunghi_dreptunghic. Să se implementeze polimorfismul pentru metodele: *citire*, *afişare*, *suprafaţa* şi *perimetru*. De la tastatură se citeşte numărul de figuri. Elaboraţi un program care va permite :

a) citirea figurilor de la tastatură;
b) afişarea acestora;
c) afişarea figurilor sub formă de dreptunghi;
d) afişarea figurilor sub formă de triunghi;
e) afişarea figurii cu suprafaţă maximă;
f) afişarea figurii cu perimetru maxim;
g) afişarea perimetrului total al figurilor;
h) afişarea suprafeţei totale a figurilor;
i) sortarea figurilor în ordine descrescătoare suprafeţei.

3. Se consideră drept bază clasa *tread* (va avea în calitate de date două numere reale şi un număr întreg). În baza acestei clase se derivă clasele paralelogram şi triunghi. Să se implementeze polimorfismul pentru metodele: *citire*, *afişare*, *suprafaţa* şi *perimetru*. De la tastatură se citeşte numărul de figuri. Elaboraţi un program care va permite :

a) citirea figurilor de la tastatură;
b) afişarea acestora;
c) afişarea figurilor sub formă de paralelogram;
d) afişarea figurilor sub formă de triunghi;
e) afişarea figurii cu suprafaţă maximă;
f) afişarea figurii cu perimetru maxim;
g) afişarea perimetrului total al figurilor;
h) afişarea suprafeţei totale a figurilor;
i) sortarea figurilor în ordine descrescătoare suprafeţei.

4. Se consideră drept bază clasa *patrulater*. În baza acestei clase se derivă clasele paralelogram şi trapez. Să se implementeze polimorfismul pentru metodele: *citire*, *afişare*, *suprafaţa* şi *perimetru*. De la tastatură se citeşte numărul de figuri. Elaboraţi un program care va permite :

a) citirea figurilor de la tastatură;
b) afişarea acestora;
c) afişarea figurilor sub formă de paralelogram;
d) afişarea figurilor sub formă de trapez;
e) afişarea figurii cu suprafaţă maximă;
f) afişarea figurii cu perimetru maxim;
g) afişarea perimetrului total al figurilor;
h) afişarea suprafeţei totale a figurilor;
i) sortarea figurilor în ordine descrescătoare suprafeţei.

5. Se consideră drept bază clasa *dreptunghi*. În baza acestei clase se derivă clasele piramidă şi prismă. Să se implementeze polimorfismul pentru metodele: *citire*, *afişare*, *suprafaţa* şi *volum*. De la tastatură se citeşte numărul de figuri. Elaboraţi un program care va permite :

a) citirea figurilor de la tastatură;
b) afişarea acestora;
c) afişarea figurilor sub formă de piramidă;
d) afişarea figurilor sub formă de prismă;
e) afişarea figurii cu suprafaţă maximă;
f) afişarea figurii cu volum maxim;
g) afişarea volumului total al figurilor;
h) afişarea suprafeţei totale a figurilor;
i) sortarea figurilor în ordine descrescătoare suprafeţei.

6. Se consideră drept bază clasa *paralelogram*. În baza acestei clase se derivă clasele piramidă şi prismă. Să se implementeze polimorfismul pentru metodele: *citire*, *afişare*, *suprafaţa* şi *volum*. De la tastatură se citeşte numărul de figuri. Elaboraţi un program care va permite :

a) citirea figurilor de la tastatură;
b) afişarea acestora;
c) afişarea figurilor sub formă de piramidă;
d) afişarea figurilor sub formă de prismă;
e) afişarea figurii cu suprafaţă maximă;
f) afişarea figurii cu volum maxim;
g) afişarea volumului total al figurilor;

h) afişarea suprafeţei totale a figurilor;
i) sortarea figurilor în ordine descrescătoare suprafeţei.

7. Se consideră drept bază clasa *cerc*. În baza acestei clase se derivă clasele con şi cilindru. Să se implementeze polimorfismul pentru metodele: *citire*, *afişare*, *suprafaţa* şi *volum*. De la tastatură se citeşte numărul de figuri. Elaboraţi un program care va permite :

a) citirea figurilor de la tastatură;
b) afişarea acestora;
c) afişarea figurilor sub formă de con;
d) afişarea figurilor sub formă de cilindru;
e) afişarea figurii cu suprafaţă maximă;
f) afişarea figurii cu volum maxim;
g) afişarea volumului total al figurilor;
h) afişarea suprafeţei totale a figurilor;
i) sortarea figurilor în ordine descrescătoare suprafeţei.

8. Se consideră ierarhia: *Persoana*, *Angajat* şi *Student*. Clasele Angajat şi Student sunt derivatele clasei Persoana. În clasa persoană vor fi definite următoarele:

date: numele, prenumele, anul naşterii;
metode:
Citire – citirea de la tastatură a datelor despre persoană,
Afisare – afişarea la consolă a datelor despre persoană;
Virsta – va returna vîrsta persoanei.

În clasa Angajat suplimentar vor mai fi declarate:

date: Ore – numărul de ore lucrate, Pl_ora – plata pentru o oră, funcţia;
metode: *Bani* – va determina salariul angajatului conform formulei: *ore*pl_ora.*

În clasa Student suplimentar vor mai fi declarate

date: media, grupa;
metode *Bani* – va determina bursa studentului conform formulei: media*75 lei, dacă media studentului este mai mare decît şapte, în caz contrar returnează 0.

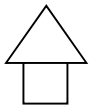
Pentru această ierarhie se va implementa polimorfismul pentru metodele citire, afişare, bani. De la tastatură se citeşte numărul total de persoane *n* (n<100). Tipurile acestora se citesc în momentul execuţiei. La ecran va fi afişat un meniu cu următoarele opţiuni:

a) Afişarea tuturor persoanelor;
b) Afişarea persoanelor angajate;
c) Afişarea persoanelor care sunt studenţi;

d) Suma totală de bani, pe care o primesc toate persoanele citite;
e) Persoana sau persoanele ce primesc cei mai mulţi bani;
f) Persoana sau persoanele ce primesc cei mai puţini bani;
g) Persoana sau persoanele cele mai tinere;
h) Afişarea persoanelor în ordine descrescătoare veniturilor acumulate;
i) Afişarea persoanelor în ordine crescătoare vîrstei.

5. Moştenire multiplă

A. *Problemă exemplu* În calitate de clase de *bază* se consideră clasele *patrat* şi *triunghi echilateral.* Aceste clase derivă clasa *figura*, formată dintr-un pătrat şi un triunghi echilateral, grafic figura se reprezintă:

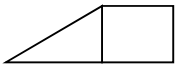


Creaţi clasa *figura*, derivata claselor *patrat* şi *triunghi echilateral*, dacă lungimea laturii triunghiului echilateral este egală cu *a+0.3*a*, unde *a* este lungimea laturii pătratului. Pentru clasa *figura* vor fi implementate metodele de determinare a suprafeţei şi a perimetrului.

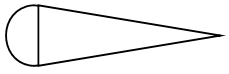
Rezolvare :

```
#include<iostream.h>
#include<math.h>
class patrat{
 public:
 double lp;
 patrat(){lp=0;}
 patrat(double a){lp=a;}
 void cit_patrat();
 void afis_patrat();
 double per_patrat(){return 4*lp;}
 double arie_patrat(){return lp*lp;}
};
void patrat::cit_patrat(){
 cout<<"Dati latura patratului"<<endl;cin>>lp;
}
void patrat::afis_patrat(){
 cout<<"Patrat: lat="<<lp<<" Aria="<<arie_patrat();
 cout<<" Perimetru="<<per_patrat()<<endl;
}
class tr_echil{
public:
 double lt;
 tr_echil(){lt=0;}
 tr_echil(double a){lt=a;}
 void cit_triunghi();
 void afis_triunghi();
```

```
 double per_triunghi(){return 3*lt;}
 double arie_triunghi(){return sqrt(3)*0.25*lt*lt;}
};
void tr_echil::cit_triunghi(){
 cout<<"Dati latura triunghilui echilateral"<<endl;cin>>lt;
}
void tr_echil::afis_triunghi(){
 cout<<"Triunghi echilateral: lat="<<lt;
 cout<<" Aria="<<arie_triunghi();
 cout<<" Perimetru="<<per_triunghi()<<endl;
}
class figura : public patrat, public tr_echil{
 public:
 figura(){lt=0;lp=0;}
 figura(double a):patrat(a),tr_echil(a+a*0.3){};
 void cit_figura();
 void afis_figura();
 double arie_figura(){return arie_triunghi()+arie_patrat();}
 double per_figura(){return per_triunghi()+per_patrat()-2*lt;}
};
void figura::cit_figura(){cit_patrat();lt=lp+lp*0.3;}
void figura::afis_figura(){
 cout<<"Figura este formata din:"<<endl;
 afis_patrat();afis_triunghi();
 cout<<"Aria figurii="<<arie_figura();
 cout<<" Perimetru figurii="<<per_figura()<<endl;
}
main(){
 figura a,b(10);
 a.afis_figura();b.afis_figura();
 a.cit_figura();a.afis_figura();
}
```

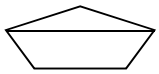
1. În calitate de clase de *bază* se consideră clasele *dreptunghi* şi *triunghi dreptunghic*. Aceste clase derivă clasa *figura*, formată dintru-un dreptunghi şi un triunghi dreptunghic, grafic figura se reprezintă: 

Creaţi clasa *figura*, derivata claselor *dreptunghi* şi *triunghi dreptunghic*, dacă lungimea catetelor triunghiului dreptunghic corespund lungimilor laturilor dreptunghiului. Pentru clasa *figura* vor fi implementate metodele de determinare a suprafeţei şi a perimetrului.

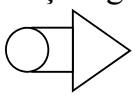
2. În calitate de clase de *bază* se consideră clasele *cerc* şi *triunghi isoscel*. Aceste clase derivă clasa *figura*, formată dintru-un cerc şi un triunghi isoscel, grafic figura se reprezintă: 

Creaţi clasa *figura*, derivata claselor *cerc* şi *triunghi isoscel*, dacă se cunoaşte lungimea razei cercului (lungimea laturii triunghiului isoscel coincide cu diametrul

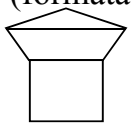
cercului) şi înălţimea triunghiului isoscel. Pentru clasa *figura* vor fi implementate metodele de determinare a suprafeţei şi a perimetrului.

3. În calitate de clase de *bază* se consideră clasele *trapez isoscel* şi *triunghi isoscel.* Aceste clase derivă clasa *figura*, formată dintru-un trapez isoscel şi un triunghi isoscel, grafic figura se reprezintă: 

Creaţi clasa *figura*, derivata claselor *trapez isoscel* şi *triunghi isoscel*, dacă se cunosc lungimile bazelor trapezului (lungimea laturii triunghiului isoscel este egală cu lungimea bazei mari a trapezului), iar înălţimile trapezului şi a triunghiului sunt de aceiaşi mărime. Pentru clasa *figura* vor fi implementate metodele de determinare a suprafeţei şi a perimetrului.

4. În calitate de clase de *bază* se consideră clasele *cerc* şi *figura* (exemplul rezolvat*).* Aceste clase derivă clasa *figura_noua* (formată dintr-un cerc şi figura prezentată în exemplul rezolvat), grafic figura_noua se reprezintă: 

Creaţi clasa *figura_noua*, derivata claselor *cerc* şi *figura*, dacă se cunoaşte lungimea laturii pătratului, lungimea diametrului cercului este egală cu lungimea laturii pătratului, iar lungimea laturii triunghiului echilateral este egală cu *a+0.3*a*, unde *a* este lungimea laturii pătratului. Pentru clasa *figura_noua* vor fi implementate metodele de determinare a suprafeţei şi a perimetrului.

5. În calitate de clase de *bază* se consideră clasele *patrat* şi *figura* (problema 3). Aceste clase derivă clasa *figura_noua* (formată dintr-un pătrat şi figura din problema 3), grafic figura_nouă se reprezintă: 

Creaţi clasa *figura_noua*, derivata claselor *figura* şi *patrat*, dacă se cunoaşte datele despre figură (problema 3), iar lungimea laturii pătratului este egală cu lungimea bazei mici a trapezului isoscel. Pentru clasa *figura_noua* vor fi implementate metodele de determinare a suprafeţei şi a perimetrului.

B. *Problemă exemplu* [2, p. 185-186].

6. Se consideră ierarhia formată din clasele:

Clasă de bază: *Punct* cu cîmpurile: coordonatele punctului şi culoarea;

Clase derivate:

triunghi echilateral (punctul se consideră a fi în centrul triunghiului): lungimea laturii;

cerc: lungimea razei;

figură (triunghi înscris în cerc) derivata claselor cerc şi triunghi.

Metodele de citire şi desenare vor fi implementate pentru fiecare clasă din ierarhie. De la tastatură se citeşte un număr întreg *n* (n<100), numărul figurilor din ierarhie. Elaboraţi un program prin intermediul căruia vor fi afişate figurile la ecran.

7. Se consideră ierarhia formată din clasele:

Clasă de bază: *Punct* cu cîmpurile: coordonatele punctului şi culoarea;

Clase derivate:

triunghi echilateral (punctul se consideră a fi în centrul triunghiului): lungimea laturii;

patrat: lungimea laturii, punctul se consideră a fi în centrul pătratului;

figură (triunghi înscris în pătrat) derivata claselor cerc şi pătrat.

Metodele de citire şi desenare vor fi implementate pentru fiecare clasă din ierarhie. De la tastatură se citeşte un număr întreg *n* (n<100), numărul figurilor din ierarhie. Elaboraţi un program prin intermediul căruia vor fi afişate figurile la ecran.

8. Se consideră ierarhia formată din clasele:

Clasă de bază: *Punct* cu cîmpurile: coordonatele punctului şi culoarea;

Clase derivate:

triunghi echilateral: lungimea laturii;

patrat: lungimea laturii, punctul se consideră a fi în centrul pătratului;

figură (o casă formată din pătrat şi triunghi) derivata claselor patrat şi triunghi.

Metodele de citire şi desenare vor fi implementate pentru fiecare clasă din ierarhie. De la tastatură se citeşte un număr întreg *n* (n<100), numărul figurilor din ierarhie. Elaboraţi un program prin intermediul căruia vor fi afişate figurile la ecran.

9. Se consideră o ierarhie de clase care să descrie habitatul animalelor dintr-o rezervaţie naturală. În rezervaţie pot fi întîlnite următoarele animale: *iepuri*, *urşi*, *căprioare*, *vulpi* şi *lei*. Unele dintre animale sunt ierbivore, altele carnivore sau chiar ierbivore şi carnivore. Ierarhia de clase va avea clasă de bază *CAnimal*, clasă abstractă, din care se vor deriva două clase: *CAnimalIerbivor* şi *CAnimalCarnivor*. Pentru fiecare tip de animal se va proiecta o clasă corespunzătoare care va extinde una din clase de mai sus, sau ambele clase, corespunzător modului de hranire a animalului. Despre un animal se cunosc următoarele informaţii comune: *nume*, *data aducerii în rezervaţie*, *greutate*, *hrana preferată*, *cantitate hrană pe zi*. În plus

despre *iepure* se cunoaşte suprafaţa minimă pe care poate trăi;

despre *vulpe* - temperatura maximă la care poate rezista, tipul (polară, indigenă);

despre *leu* – temperatura minimă suportată, ţara de proveniență;

despre *urs* - perioada de hibernare;

despre *căprioară* - numărul de pui.

Pentru fiecare clasa se vor implementa atît o funcţie de citire a datelor, cît şi o funcţie de afişare a acestora, funcţii ce vor fi declarate ca funcţii pur virtuale în cadrul clasei de bază *CAnimal*. Creaţi clasa *CRezervatie* care să conţină o mulţime de animale şi să se populeze rezervaţia.

10. Se consideră ierarhia: *Persoana*, *Angajat*, *Student* şi *Student_angajat*. Clasele Angajat şi Student sunt derivatele clasei Persoana, iar clasa Student_angajat este derivata claselor Student şi Angajat. În clasa persoană vor fi definite următoarele:

date: numele, prenumele, anul naşterii;

metode:

Citire – citirea de la tastatură a datelor despre persoană,

Afisare – afişarea la consolă a datelor despre persoană;

Virsta – va returna vîrsta persoanei.

În clasa Angajat suplimentar vor mai fi declarate:

date: Ore – numărul de ore lucrate, Pl_ora – plata pentru o oră, funcţia;

metode: *Bani* – va determina salariul angajatului conform formulei: *ore*pl_ora.*

În clasa Student suplimentar vor mai fi declarate

date: media, grupa;

metode *Bani* – va determina bursa studentului conform formulei: media*75 lei, dacă media studentului este mai mare decît şapte, în caz contrar returnează 0.

Clasa Student_angajat va conţine metode pentru *citire*, *afişare*, *Bani (salariu +bursa)*.

Pentru această ierarhie se va implementa polimorfismul pentru metodele citire, afişare, bani. De la tastatură se citeşte numărul total de persoane *n* (n<100). Tipul acesteia se citeşte în momentul execuţiei. La ecran va fi afişat un meniu cu următoarele opţiuni:

a) Afişarea tuturor persoanelor;
b) Afişarea persoanelor grupate pe categorii;
c) Suma totală de bani, pe care o primesc toate persoanele citite;
d) Persoana sau persoanele ce primesc cei mai mulţi bani;
e) Persoana sau persoanele ce primesc cei mai puţini bani;
f) Afişarea persoanelor în ordine descrescătoare veniturilor acumulate;
g) Afişarea persoanelor în ordine crescătoare vîrstei.

6. Supraîncărcarea operatorilor

Problemă exemplu. [2, p. 164-168].

1. Se consideră clasa *timp*, cu următoarea structură:

```
class timp{
 int ora,min, sec;//14:24:35
 public:
 timp();
 timp(int,int,int);
 //…
};
```

Elaboraţi un program, prin intermediul căruia pentru clasa *timp* vor fi supraîncărcaţi:

a) Operatorul + va efectua suma dintre două obiecte de tipul timp;
b) Operatorul - va efectua diferenţa dintre două obiecte de tipul timp;
c) Operatorul ++ incrementarea cu o secundă;
d) Operatorul -- decrementarea cu o secundă;
e) Operatorul =
f) Operatorii >>,<< vor citi/afişa timpul în formatul *ora:min:sec*, de ex. 14:24:35;
g) Operatorii de comparaţie ==,!=, >, >=, <,<=;
h) Se va realiza o metodă, care va prelucra timpul în forma corectă, de ex. dacă timpul va fi 2:59:63, va fi transformat în 3:00:03.

2. Se consideră clasa *data*, cu următoarea structură:

```
class data{/*se consideră că o luna are exact 30 de zile*/
 int an,luna, zi;//2010:11:18
 public:
 data();
 data(int,int,int);
 //…
};
```

Elaboraţi un program, prin intermediul căruia pentru clasa *data*, vor fi supraîncărcaţi:

a) Operatorul + va efectua suma dintre două obiecte de tipul data;
b) Operatorul - va efectua diferenţa dintre două obiecte de tipul data;
c) Operatorul ++ incrementarea cu o zi;
d) Operatorul -- decrementarea cu o zi;
e) Operatorul =
f) Operatorii >>,<< vor citi/afişa data în formatul *an:luna:zi*, de ex. 2010:11:18;
g) Operatorii de comparaţie ==,!=, >, >=, <,<=;
h) Se va realiza o metodă, care va prelucra data în forma corectă, de ex. dacă data va fi 2010:12:32, va fi transformat în 2011:01:02.

3. Se consideră clasa *fractie*, cu următoarea structură:

```
class fractie{
 public:
 int numitor,numarator;// 14/35
 fractie ();
 fractie (int,int);
 //…
};
```

Elaboraţi un program prin intermediul căruia pentru clasa *fractie*, vor fi supraîncărcaţi:

a) Operatorii aritmetici +, -, *, / ;
b) Operatorii incrementare şi decrementare ++, --;
c) Operatorul =;
d) Operatorii >>,<< vor citi/afişa fracţia în formatul numărător/numitor, de ex. 14/35;
e) Operatorii de comparaţie ==,!=, >, >=, <,<=;
f) Se va realiza o metodă care va transforma fracţia în fracţie ireductibilă.

4. Se consideră clasa *multime*, cu următoarea structură:

```
class multime{
 char *m;/* tabloul in care vor fi pastrate elementele
multimii*/
 int k;// cardinalul multimii
 public:
 multime ();//va aloca dinamic 255 de elemente
 multime (int);//se transmite cardinalul multimii
 //…
};
```

Elaboraţi un program, prin intermediul căruia pentru clasa *multime*, vor fi supraîncărcaţi:

a) Operatorul + va efectua operaţia de reuniune a mulţimilor;
b) Operatorul - va efectua diferenţa mulţimilor;
c) Operatorul * va efectua operaţia de intersecţie a mulţimilor;
d) Operatorul =;
e) Operatorii <= şi >= vor verifica incluziunea mulţimilor;
f) Operatorii >>,<< vor citi/afişa elementele mulţimii;
g) Se va implementa metoda `int cauta(const char);`

care, va returna 1 dacă elementul transmis ca parametru este inclus în mulţime şi 0 în caz contrar.

5. Se consideră clasa *String* cu următoarea structură:

```
class string{
 char *s
```

```
 int k;// lungimea sirului
 public:
 string ();//va aloca dinamic 255 de elemente
 string (int);//se transmite lungimea sirului
 int ToInt(string);
 string ToString(int);
 int cauta(char);
 //…
};
```

Elaboraţi un program, prin intermediul căruia pentru clasa *string*, vor fi supraîncărcaţi:

a) Operatorul + va concatena două şiruri;
b) Operatorul =;
c) Operatorii de comparaţie ==,!=, >, >=, <,<=;
d) Operatorii >>,<< vor citi/afişa elementele şirului;
e) Metoda *cauta*, va returna 1 dacă elementul transmis ca parametru este inclus în şir şi 0 în caz contrar.
f) Metoda *ToInt*, va transforma şirul înt-un număr întreg şi va returna acest număr, în cazul în care nu este posibil de efectuat transformarea se va returna valoarea 0.
g) Metoda *ToString*, va transforma un număr întreg în şir.

6. Se consideră clasa *Natural*, cu următoarea structură:

```
class Natural{
 public:
 int t[100];//tablou ce conţine cifrele numarului
 Natural ();//va zerografia tabloul t
 //…
};
```

Elaboraţi un program, prin intermediul căruia pentru clasa *Natural*, vor fi supraîncărcaţi:

a) Operatorii aritmetici +, -, *, /, % , vor efectua calcule cu numere mari, de pînă la 100 cifre;
b) Operatorii incrementare şi decrementare ++, --;
c) Operatorul =;
d) Operatorii >>,<< vor citi/afişa numărul la ecran;
e) Operatorii de comparaţie ==,!=, >, >=, <,<=.

7. Se consideră clasa *Trinom*, cu următoarea structură:

```
class Trinom{
 double a,b,c;// ax^2+bx+c
 public:
 Trinom ();//va initializa coefienţii a,b,c cu 0
```

```
 Trinom (double, double, double);
 void Radacini();
 //…
};
```

Elaboraţi un program, prin intermediul căruia pentru clasa *trinom*, vor fi supraîncărcaţi:

a) Operatorul + va efectua operaţia de adunare dintre două trinoame;
b) Operatorul - va efectua operaţia de scădere dintre două trinoame;
c) Operatorul =;
d) Operatorii de comparaţie ==,!=, >, >=, <,<=;
e) Operatorii >>,<< vor citi/afişa polinomul la ecran;
f) Metoda *Radacini*, va afişa rădăcinile reale ale trinomului.

8. Se consideră clasa *Binar*, cu următoarea structură:

```
class Binar{
 public:
 int t[1000];
 Binar();//va zerografia tabloul t
 int ToInt();
 Binar ToBinar(int);
 //…
};
```

Elaboraţi un program, prin intermediul căruia pentru clasa *Binar* vor fi supraîncărcaţi:

a) Operatorii aritmetici +, -, vor determina suma/diferenţa dintre două numere binare;
b) Operatorii incrementare şi decrementare ++, --;
c) Operatorul =;
d) Operatorii >>,<< vor citi/afişa numărul la ecran numărul binar;
e) Operatorii de comparaţie ==,!=, >, >=, <,<=;
f) Metoda *ToInt*, va transforma numărul binar în baza 10.
g) Metoda *ToBinar*, va transforma un număr întreg, din baza 10 în baza doi.

7. Agregare

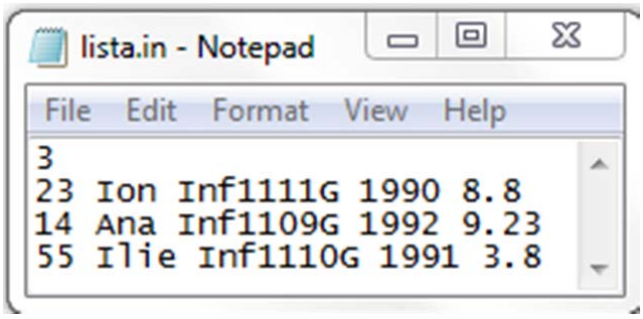
Este necesar de a elabora un program prin intermediul căruia se va gestiona o listă. În acest caz lista reprezintă studenţii unei grupe. În toate sarcinile trebuie să fie prezente:

a) constructor – crearea listei vide;
b) destructor – va elibera memoria Heap de obiecte;
c) metoda de creare – va încărca datele în Heap din fişierul de intrare;
d) metoda de inserare – va insera un obiect în Heap;

e) metoda de excludere – va exclude un obiect din Heap;
f) metoda de căutare –va fi supraîncărcată, astfel încît căutarea se va efectua după un parametru şi mai mulţi parametri;
g) metode pentru filtrarea datelor;
h) două metode de sortare, una după un cîmp cu valori numerice, alta după un cîmp text;
i) un meniu, prin intermediul căruia utilizatorul va avea posibilitatea de a alege operaţia dorită.

Realizare:

Creăm fişierul *lista.in* şi îl salvăm în directoriul curent. În fişier datele vor fi scrise în mod corespunzător figurii:




```
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
class TStudent{
 char *nume,*grupa;
 int anul,id;
 double media;
 TStudent *next;//adresa umatorului element
 TStudent();
 friend istream & operator>>(istream &, TStudent *&);
 friend ostream & operator<<(ostream &, TStudent*);
 friend class lista;
 friend int schimb(TStudent *,TStudent *);
};
TStudent::TStudent(){
 nume=new char[20];grupa=new char[10];
 anul=0;id=0;media=0; next=NULL;
}
istream & operator>>(istream &is, TStudent *&t){
 cout<<"Dati datele despre student"<<endl;
 cout<<"Id: ";is>>t->id;
 cout<<"Nume: ";is>>t->nume;
 cout<<"Grupa ";is>>t->grupa;
 cout<<"Anul: ";is>>t->anul;
 cout<<"Media: ";is>>t->media;
 return is;
}
```

```
ostream & operator<<(ostream &os, TStudent *t){
 os<<setw(6)<<t->id<<setw(20)<<t->nume<<setw(10);
 os<<t->grupa<<setw(6)<<t->anul<<setw(7)<<t->media<<endl;
 return os;
}
int schimb(TStudent *a, TStudent *b){
 TStudent *an,*bn,t;
 if (!a || !b)return 0;
 an=a->next;bn=b->next;t=*a;*a=*b;*b=t;a->next=an;b->next=bn;
 return 1;
}
class lista{
 TStudent *prim;//adresa primului element din lista
 int n;//numarul total de studenti din lista
 public:
 lista(){prim=NULL;n=0;}
 void creare();
 void afisare();
 void inserare();
 void exclude();
 void cauta(char *);
 void cauta(double,int);
 int sort_medie();
 ~lista();
};
void lista::creare(){
 TStudent *p; ifstream f("lista.in");f>>n;
 for(int i=0;i<n;i++){
  p=new TStudent;
  f>>p->id>>p->nume>>p->grupa>>p->anul>>p->media;
  p->next=prim;prim=p;
 }f.close();
}
void lista::afisare(){
 cout<<"Lista studentilor"<<endl;
 cout<<setw(6)<<"id"<<setw(20)<<"nume"<<setw(10)<<"grupa";
 cout<<setw(6)<<"anul"<<setw(7)<<"media"<<endl;
 TStudent *p;p=prim;
 while(p!=NULL){cout<<p;p=p->next;}
}
void lista::inserare(){
 TStudent *q;q=new TStudent;
 cin>>q; q->next=prim;prim=q; n++;
}
void lista::exclude(){
 TStudent *p,*q;int c;
 cout<<"Introdu id-ul studentului care va fi exclus";
 cin>>c; p=prim;
 if(p->id==c){prim=p->next;delete p;n--;}
 else{
  while(p->next->id!=c && p!=NULL)p=p->next;
  if(p!=NULL){q=p->next;p->next=q->next;delete q;n--;}
 }
}
```

```
void lista::cauta(char *nume){
 TStudent *p;p=prim;
 while(p!=NULL){
  if(stricmp(p->nume,nume)==0) cout<<p;p=p->next;
 }
}
void lista::cauta(double media, int anul){
 TStudent *p;p=prim;
 while(p!=NULL){
  if(p->media>=media&&p->anul==anul) cout<<p;p=p->next;
 }
}
int lista::sort_medie(){
 int i,j;TStudent *a,*b;
 if(!prim)return 0;
 a=prim;
 for(i=0;i<n-1;i++){
  b=a->next;
  for(j=i+1;j<n;j++){
   if((a->media)>(b->media))schimb(a,b);
   b=b->next;
  }a=a->next;
 }
 clrscr();
 cout<<"Sortarea datelor s-a efectuat cu succes";getch();
 return 1;
}
lista::~lista(){
 TStudent *p,*q;
 do{
  p=prim;prim=prim->next;delete p;
 }while(prim!=NULL);
 }
main(){
 char c;lista g;
 g.creare();
 do{
  clrscr();
  cout<<"Tastati"<<endl<<"1-afisare"<<endl<<"2-excludere";
  cout<<endl<<"3-inserare"<<endl<<"4-Studentii cu numele Ion";
  cout<<endl<<"5-Studentii cu media>8, nascuti in anul 1990";
  cout<<endl<<"6-Sortare dupa medie"<<endl<<"0-iesire"<<endl;
  c=getch(); clrscr();
  switch(c){
   case '1': g.afisare();getch();break;
   case '2': g.exclude();break;
   case '3': g.inserare();break;
   case '4': g.cauta("Ion");getch();break;
   case '5': g.cauta(8,1990);getch();break;
   case '6': g.sort_medie();break;
  }
 }while(c!='0');
}
```

1. Elaboraţi un program prin intermediul căruia vor fi gestionate filmele de la un cinematograf. Despre un film se cunosc următoarele date: numele filmului, anul ediţiei, actorul principal, regizorul, bugetul filmului, ora de difuzare a filmului în cinematograf.

2. Elaboraţi un program prin intermediul căruia vor fi gestionate mai multe state prezente la un concurs. Despre fiecare stat se cunosc următoarele date: numele ţării, preşedintele, suprafaţa, numărul de locuitori, continentul.

3. Elaboraţi un program prin intermediul căruia vor fi gestionate cărţile unei biblioteci. Despre fiecare carte se cunosc următoarele date: denumirea cărţii, autorul, editura, numărul de pagini, anul ediţiei, preţul.

4. Elaboraţi un program prin intermediul căruia vor fi gestionate calculatoarele unei universităţi. Despre un calculator se cunosc următoarele date: tipul monitorului, capacitatea procesorului, cantitatea memoriei operative, placa video, hard disk-ul.

5. Elaboraţi un program prin intermediul căruia vor fi gestionate telefoanele mobile ale unui magazin. Despre un telefon se cunosc următoarele date: marca, anul ediţiei, culoarea, preţul, termenul de garanţie, tipul camerei video.

6. Elaboraţi un program prin intermediul căruia vor fi administrate hotelurile dintr-o ţară. Despre un hotel se cunosc următoarele date: nume hotel, adresa, numărul de stele, numărul de odăi (camere), costurile minim şi maxim pentru o cameră.

7. Elaboraţi un program prin intermediul căruia vor fi gestionate automobilele dintr-un salon auto. Despre un automobil se cunosc următoarele date: marca, anul ediţiei, culoarea, preţul, capacitatea motorului, consumul la 100 km.

8. Creaţi un tip de date *matr_complex*, care va fi o matrice de numere complexe. Asupra acestui tip de date vor fi supraîncărcaţi toţi operatorii specifici unei matrice.

9. Elaboraţi un program prin intermediul căruia se va crea tipul de date tablou bidimensional cu linii de lungime variabilă, notat **TBV**. Pentru obiectul de tip **TBV** se vor implementa metodele:

a) *citire()* – citirea elementelor obiectului;
b) *afisare()* – afișarea elementelor obiectului;
c) *minime()* – afișarea elementelor minime de pe fiecare linie;

d) *maxime()* – afişarea elementelor maxime de pe fiecare linie;
e) *sum_max()* – determinarea sumei elementelor maxime de pe fiecare linie;
f) *sum_min()* – determinarea sumei elementelor minime de pe fiecare linie;
g) *cauta(int e)* – va returna valoare 1 dacă elementul *e* există şi 0 în caz contrar;
h) *sort()* – sortarea în ordine crescătoare pe linii a obiectului de tip TBV;
i) **sort([char mod],[int linie])* – sortarea cu transmitere de parametri c (ordine crescătoare) sau d (ordine descrescătoare) în calitate de mod şi cu indicarea liniei.

10. Elaboraţi un program prin intermediul căruia se va efectua evidenţa personalului dintr-o instituţie. În instituţia dată sunt următoarele tipuri de personal :
Angajat – despre care se cunosc datele: numele, prenumele, codul fiscal, numărul de ore lucrate, plata pentru o oră.

În calitate de metode vor fi: *citirea* datelor de la tastatură, afişarea datelor la ecran, determinarea salariului conform formulei: *salariu=ore_lucrate*plata_ora*.
Angajat de bază – despre care se cunosc datele: numele, prenumele, codul fiscal, numărul de ore lucrate, plata pentru o oră, gradul (0,1,2,3), anul angajării.

În calitate de metode vor fi: *citirea* datelor de la tastatură, afişarea datelor la ecran, determinarea salariului conform formulei: *salariu=ore_lucrate*plata_ora*. Angajatul va beneficia şi de un adaos la salariu. Salariaţii cu gradul 0 vor avea un adaos de 50%, gradul 1 – 40%, gradul 2 – 30%. În dependenţă de stagiu adaosul va fi de 35% pentru cei cu un stagiu ≥ 10 ani, 25% pentru cei cu un stagiu ≥ 3 ani şi <10 ani şi 10% pentru toţi ceilalţi.
Student – despre care se cunosc datele: numele, prenumele, codul fiscal, grupa, media.

În calitate de metode vor fi: *citirea* datelor de la tastatură, afişarea datelor la ecran, determinarea bursei. Bursa studentului o vom determina astfel:

$$Bursa = \begin{cases} 0\ \ lei, dacă\ \ media < 7.5 \\ 300\ \ lei, dacă\ \ 7.5 \le media < 8.5 \\ 400\ \ lei, dacă\ \ 8.5 \le media < 9.5 \\ 500\ \ lei, dacă\ \ media \ge 9.5 \end{cases}$$

Student Angajat – despre care se cunosc datele: numele, prenumele, codul fiscal, grupa, media, numărul de ore lucrate, plata pentru o oră.

În calitate de metode vor fi: *citirea* datelor de la tastatură, afişarea datelor la ecran, determinarea bursei şi a salariului. Metodele bursa şi salariu vor efectua calculele conform formulelor din clasele precedente.

Asupra personalului vor fi efectuate operaţii de adăugare a unei persoane, excludere, afişare a tuturor persoanelor din instituţie, cît şi determinarea bugetului lunar de salarizare a personalului instituţiei.

Capitolul II Mediul de programare vizuală C++ Builder

C++Builder este un mediu de dezvoltare rapidă a aplicaţiilor produs de filiala CodeGear a Embarcadero Technologies pentru scrierea programelor în limbajul de programare C++. Prima versiune datează anului 1997. Se presupune că o versiune viitoare a CodeGear C++Builder va avea suport pentru x86/64 şi va crea cod nativ x86/64.

Programele au fost elaborate în versiunea C++ Builder 6.

Probleme rezolvate

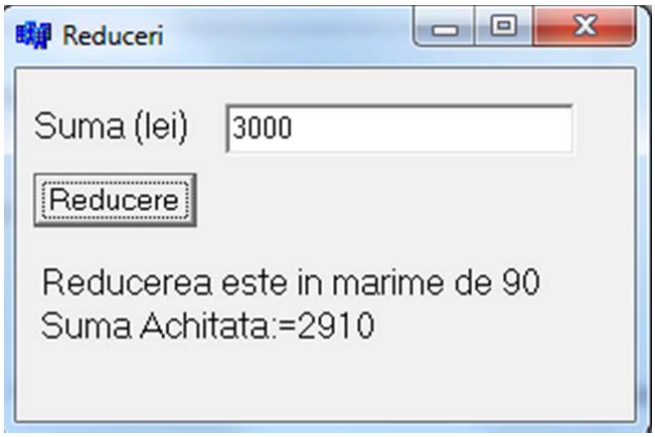
Problema1 Reduceri

Elaboraţi o aplicaţie care va determina reducerea unui produs cumpărat. Cumpărătorul beneficiază de reducere în următoarele cazuri :

a) în cazul în care suma este mai mare ca 1000 lei reducerea este de 3%;

b) în zilele de odihnă reducerea este de 2%.

În rezultatul efectuării unui click pe butonul *Reducere*, în componenta Label va fi afişată informaţia completă despre reducere cu indicarea sumei finale. În cazul în care reduceri nu sunt, va fi afişat mesajul corespunzător. Informaţia despre zilele de odihnă va fi determinată în baza analizei datei curente. Forma aplicaţiei va arăta astfel :



Realizare:

Pe suprafaţa formei plasăm componentele:

Label1 – modificăm proprietatea Caption în *Suma (lei)*;

Label2 – eliminăm textul de la proprietatea Caption şi setăm dimensiunile corespunzător;

Edit1 - eliminăm textul de la proprietatea Text;

Button1 – modificăm proprietatea Caption în *Reducere*;
Pentru componenta *Form1* setăm proprietăţile:

Caption=Reduceri

Position= poScreenCenter

Width=300

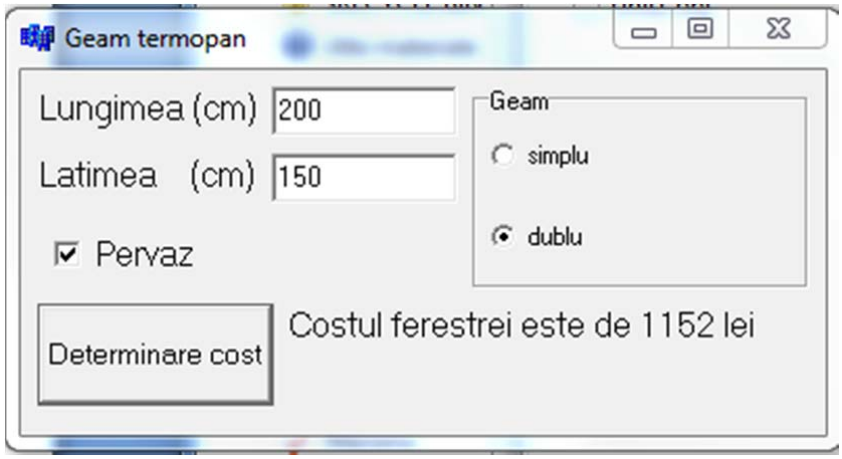
Height=200

Prelucrăm evenimentul Button1Click:

```
double suma, reducere, ziua;
 TDateTime data=Now();//stabilim timpul curent
 suma=Edit1->Text.ToDouble();
 if(suma>1000) reducere=suma*0.03;ziua=DayOfWeek(data);
 if(ziua==7||ziua==1) reducere+=suma*0.02;
 Label2->Caption="Reducerea este in marime de "
 +AnsiString(reducere)+'\n'+
 "Suma Achitata:="+AnsiString(suma-reducere);
```

<u>Problema2</u> Termopan

Elaboraţi o aplicaţie, prin intermediul căreia va fi posibil de determinat costul unui geam termopan. Preţurile sunt: $1m^2$ costă 179 lei, pervazul ferestrei costă 39 lei/m. Aplicaţia va arăta astfel :



Realizare:

Pe suprafaţa formei plasăm componentele:

Label1 – modificăm proprietatea Caption în *Lungimea (cm)*;

Label2– modificăm proprietatea Caption în *Latimea (cm)*;

Label3 – eliminăm textul de la proprietatea Caption şi setăm dimensiunile corespunzător;

Edit1 - eliminăm textul de la proprietatea Text;

Edit2 - eliminăm textul de la proprietatea Text;

Button1 – modificăm proprietatea Caption în *Determinare cost*;

CheckBox1 - modificăm proprietatea Caption în *Pervaz*;

RadioGroup1 - modificăm proprietatea Caption în *Geam*, iar prin intermediul proprietăţii Items introducem *simplu* şi *dublu*;

Pentru componenta *Form1* setăm proprietăţile:

Caption=Geam termopan

Position= poScreenCenter

Width=400

Height=210

Prelucrăm evenimentul Button1Click:

```
double lung, lat, cost;
bool pervaz=CheckBox1->Checked;
lung=Edit1->Text.ToDouble()*0.01;
lat=Edit2->Text.ToDouble()*0.01;
cost=lung*lat*179;
if(RadioGroup1->ItemIndex==1) cost=cost*2;
if(pervaz)cost+=lung*39;Label3->Caption="Costul ferestrei este de
"+ AnsiString(cost)+" lei";
```

Problema3 Navigator

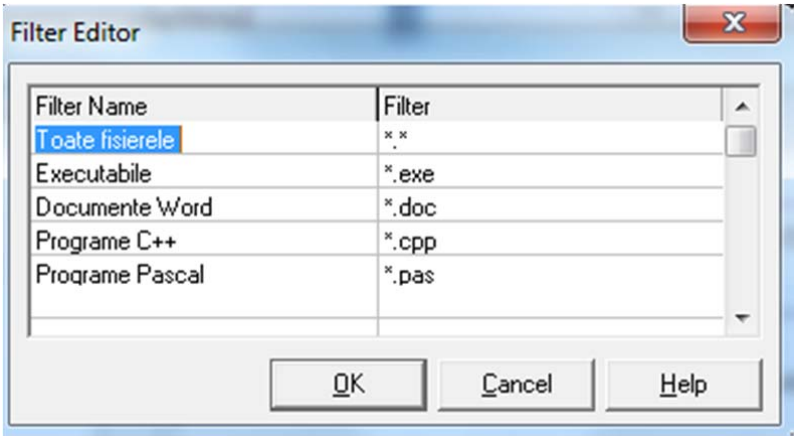
Elaboraţi o aplicaţie prin intermediul căreia va fi efectuată navigarea în sistemul de fişiere al calculatorului. Efectuînd operaţia double-click pe numele fişierului, acesta va fi deschis. Prin intermediul unui buton cu titlul *Cauta* se va efectua căutarea fişierelor după nume, iar datele despre acestea vor fi adăugate într-o componentă de tip TMemo. Componentei de tip TMemo îi va fi ataşat un meniu contextual cu titlul *Salveaza*, efectuarea unui click pe suprafaţa acestuia va salva căutarea în fişierul *date.txt* aflat în directoriul curent.

Realizare:

Pe suprafaţa formei plasăm corespunzător componentele:

FileListBox1 – pentru afişarea fişierelor;

FilterComboBox1 – modificăm proprietatea FileList în FileListBox1, prin intermediul proprietăţii filter creăm următorul filtru:



DirectoryListoBox1 – modificăm proprietatea DirList în FilelistBox1;

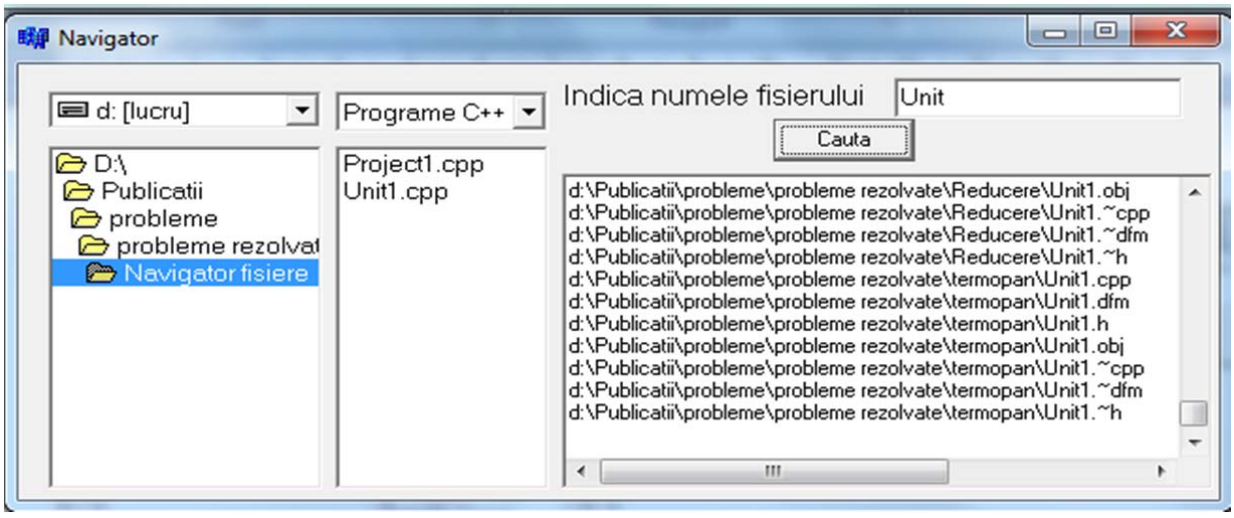
DriveComboBox1 – modificăm proprietatea FileList în DirectoryListoBox1;

Label1 – modificăm proprietatea Caption în *Indica numele fisierului*;

Edit1 - eliminăm textul de la proprietatea Text;

Button1 – modificăm proprietatea Caption în *Cauta*;
PopupMenu1 – double-click pe subcomponentă şi scriem textul *Salveaza*;
Memo1 – modificăm proprietatea PopupMenu în PopupMenu1, iar ScollBars=ssBoth;

Forma aplicaţiei va arăta astfel :



Pentru căutare vom scrie în antetul fişierului Unit1.cpp funcţia:

```
void ListFiles(AnsiString path, TStrings* List){
 TSearchRec sr;
 AnsiString nume=Form1->Edit1->Text;
 if(FindFirst(path+"*.*", faAnyFile, sr) == 0){
  do{if (sr.Attr & faDirectory){
    if(sr.Name!=".") if (sr.Name!=".."){
     ListFiles(path+sr.Name+"\\",List);
    }    }else{AnsiString adresa=sr.Name;
     if( AnsiPos(nume,adresa)>0) List->Add(path+sr.Name);
   }}while (FindNext(sr) == 0); FindClose(sr);}}
```

Prelucrăm evenimentul *FileListBox1DblClick*

```
AnsiString str=FileListBox1->Directory+"\\";
str=str+FileListBox1->Items->Strings[FileListBox1->ItemIndex];
ShellExecute( Sender,"open",str.c_str(),NULL,NULL,1);
```

Prelucrăm evenimentul *Button1Click*

```
AnsiString disc=AnsiString(DriveComboBox1->Drive);
ListFiles(disc+":\\",Memo1->Lines);
```

Prelucrăm evenimentul *Salveaza1Click*

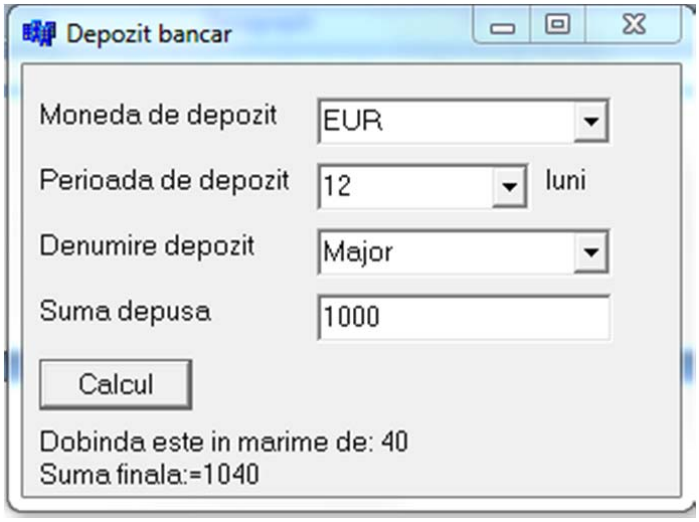
```
Memo1->Lines->SaveToFile("date.txt");
```

Problema4 Depozit bancar

Elaboraţi o aplicaţie care va permite determinarea profitului în urma depunerii unei sume de bani. Banca oferă următoarele tipuri de depozite :

DEPOZITE ÎN MDL			
Tipul depozitului	**Termenul de plasare**	**Rata dobînzii anuala**	**Retrageri din cont**
Flexibil	3 luni	8%	DA
	6 luni	9%	
	12 luni	10%	
	24 luni	11%	
	36 luni	12%	
Rentier	3 luni	9%	NU
	6 luni	10%	
	12 luni	11%	
	24 luni	12%	
	36 luni	13%	
DEPOZITE ÎN USD ŞI EUR			
Exclusiv	3 luni	2%	DA
	6 luni	2,5%	
	12 luni	3%	
	24 luni	3,5%	
	36 luni	4%	
Major	3 luni	3%	NU
	6 luni	3,5%	
	12 luni	4%	
	24 luni	4,5%	
	36 luni	5%	

Forma aplicaţiei va arăta astfel :



Realizare:

Pe suprafaţa formei plasăm componentele corespunzător:

ComboBox1 – pentru selectarea valutei, prin intermediul proprietăţii Items introducem valutele existente: MDL, USD, EUR;

ComboBox2 – pentru selectarea termenului de depozit, prin intermediul proprietăţii

Items introducem valorile: 3, 6, 12, 24, 36;
ComboBox3 – pentru selectarea numelui depozitului;
Edit1– pentru indicarea sumei depuse;
Button1 – pentru determinarea dobînzii;
6 componente de tip *TLabel* corespunzător formei aplicaţiei.

Declaraţii:

```
AnsiString denumire;
int termen;
struct depozit{
   AnsiString tip;
   double rata[5];
  }dep[4];
```

Elaborăm funcţia ce returnează indocile de ordine al termenului de depozit:

```
int k(int w){
 int c;
 switch(w){
  case 3: c=0;break;
  case 6: c=1;break;
  case 12: c=2;break;
  case 24: c=3;break;
  case 36: c=4;break;
 }
 return c;
}
```

Prelucrăm evenimentul *FormCreate*

```
dep[0].tip="Flexibil";
for(int i=0;i<5;i++)dep[0].rata[i]=8+i;
dep[1].tip="Rentier";
for(int i=0;i<5;i++)dep[1].rata[i]=9+i;
dep[2].tip="Exclusiv";
for(int i=0;i<5;i++)dep[2].rata[i]=2+i*0.5;
dep[3].tip="Major";
for(int i=0;i<5;i++)dep[3].rata[i]=3+i*0.5;
```

Prelucrăm evenimentul *ComboBox1Change*

```
ComboBox3->Clear();
int n=ComboBox1->ItemIndex;
if(n!=-1)
if(n==0){ComboBox3->Items->Add("Flexibil");
ComboBox3->Items->Add("Rentier");}
else {ComboBox3->Items->Add("Exclusiv");
ComboBox3->Items->Add("Major");}
```

Prelucrăm evenimentul *Button1Click*

```
double suma, dobinda;
suma=Edit1->Text.ToDouble();
for(int i=0;i<4;i++)
if(denumire==dep[i].tip)
dobinda=suma*dep[i].rata[k(termen)]*(termen/12.0)*0.01;
Label6->Caption="Dobinda este in marime de: "+
FormatFloat("#.###",dobinda)+
'\n'+"Suma finala:="+FormatFloat("#.###",suma+dobinda);
```

Prelucrăm evenimentul *ComboBox3Change*

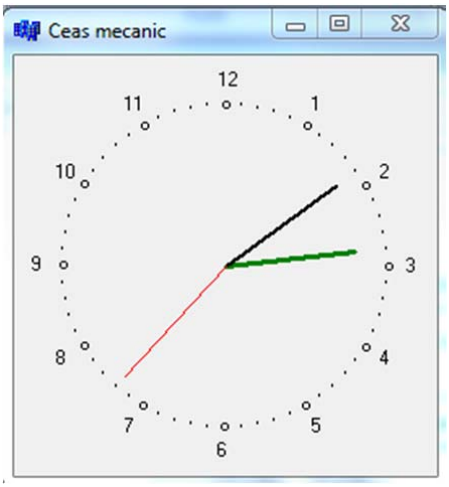
```
int n=ComboBox3->ItemIndex;
if(n!=-1)denumire=ComboBox3->Items->Strings[n];
```

Prelucrăm evenimentul *ComboBox2Change*

```
int n=ComboBox2->ItemIndex;
if(n!=-1)termen=ComboBox2->Items->Strings[n].ToInt();
```

Problema5 Ceas Mecanic [14, p.104-109]

Elaboraţi o aplicaţie prin intermediul căreia va fi simulat un ceas mecanic. Forma aplicaţiei va arăta astfel:



Realizare:

Pe suprafaţa formei plasăm componenta *Timer1*:

Declaraţii:

```
#include <vcl.h>
#pragma hdrstop
#define R 100
#define rad 0.0174533//unghiul de rotire in radiani
#include "Unit1.h"
#include "DateUtils.hpp"
#include "math.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int x0,y0,ahr,amin,asec;
```

Prelucrăm evenimentul *FormCreate*

```
TDateTime t=Now();
ClientHeight=(R+30)*2;
ClientWidth=(R+30)*2;
Form1->Caption="Ceas mecanic";
x0=R+30;y0=R+30;
ahr=90-HourOf(t)*30-(MinuteOf(Today())/12)*6;
amin=90-MinuteOf(t)*6;
asec=90-SecondOf(Today())*6;
Timer1->Interval=1000;
Timer1->Enabled=true;
```

Elaborăm funcţia:

```
void vector(int x0,int y0, int a, int l){
 int x,y;
 Form1->Canvas->MoveTo(x0,y0);
 x=x0+l*cos(a*rad);
 y=y0-l*sin(a*rad);
 Form1->Canvas->LineTo(x,y);
}
```

Elaborăm funcţia:

```
void desen(){
 TDateTime t;
 Form1->Canvas->Pen->Color=clBtnFace;
 Form1->Canvas->Pen->Width=3;
 vector(x0,y0,ahr,R-20);
 vector(x0,y0,amin,R-15);
 vector(x0,y0,asec,R-7);
 t=Now();
 ahr=90-HourOf(t)*30-(MinuteOf(t)%12)*6;
 amin=90-MinuteOf(t)*6;
 asec=90-SecondOf(t)*6;
 Form1->Canvas->Pen->Width=3;
 Form1->Canvas->Pen->Color=clGreen;
 vector(x0,y0,ahr,R-20);
 Form1->Canvas->Pen->Width=2;
 Form1->Canvas->Pen->Color=clBlack;
 vector(x0,y0,amin,R-15);
 Form1->Canvas->Pen->Width=1;
 Form1->Canvas->Pen->Color=clRed;
 vector(x0,y0,asec,R-7);
}
```

Prelucrăm evenimentul *FormPaint*

```
int x,y,a=0,h=3,pw;
TBrushStyle bs;
TColor pc;
bs=Canvas->Brush->Style;
pc=Canvas->Pen->Color;
pw=Canvas->Pen->Width;
Canvas->Brush->Style=bsClear;
Canvas->Pen->Width=1;
Canvas->Pen->Color=clBlack;
while(a<360){
 x=x0+R*cos(a*rad);
 y=x0-R*sin(a*rad);
 Canvas->MoveTo(x,y);
 if((a%30)==0){
  Canvas->Ellipse(x-2,y-2,x+3,y+3);
  x=x0+(R+15)*cos(a*rad);
  y=x0-(R+15)*sin(a*rad);
  Canvas->TextOutA(x-5,y-7,IntToStr(h));
  h--;
  if(h==0)h=12;
 }
 else Canvas->Ellipse(x-1,y-1,x+1,y+1);
 a=a+6;
```

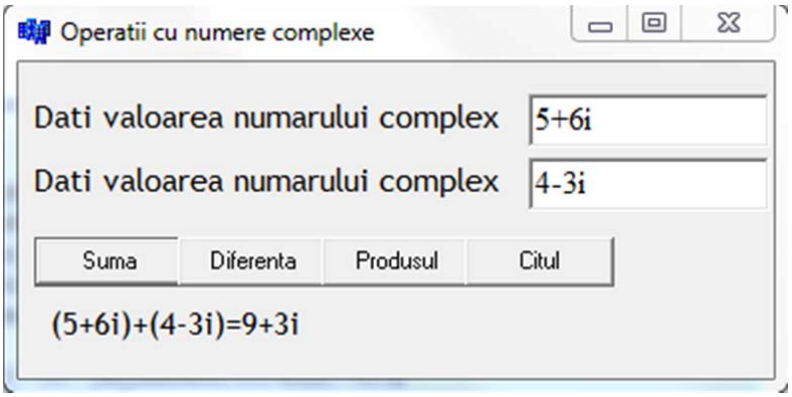
```
}
Canvas->Brush->Style=bs;
Canvas->Pen->Color=pc;
Canvas->Pen->Width=pw;
desen();
```

Prelucrăm evenimentul *Timer1Timer*

```
desen();
```

Problema6 Calculator pentru numere complexe

Elaboraţi o aplicaţie prin intermediul căreia se va efectua operaţii cu numere complexe. Forma aplicaţiei va arăta astfel:



Realizare:

Pe suprafaţa formei plasăm corespunzător componentele:

Edit1, Edit2 – pentru scrierea numerelor complexe, vom seta proprietatea *Text* cu şir vid;

Button1, Button2, Button3, Button4 – butoane pentru efectuarea calculelor cu numere complexe, setăm proprietatea Caption a fiecărui buton, în mod corespunzător: Suma, Diferenta, Produsul, Citul;

Label1, Label2 – conform figurii;

Label3 – pentru afişarea rezultatului, vom seta proprietatea *Caption* cu şir vid;

Creăm tipul de date complex:

```
class complex{
 public:
 double x,y;//complex=x+yi
 complex(){x=y=0.0;}
 complex(double a,double b){x=a;y=b;}
 complex(AnsiString s);
 friend complex operator+(complex,complex);
 friend complex operator-(complex,complex);
 friend complex operator*(complex,complex);
 friend complex operator/(complex,complex);
 complex operator=(complex b){x=b.x;y=b.y;return *this;}
 AnsiString ToStr();
 friend complex ToComplex(AnsiString);
};
```

```
complex::complex(AnsiString s){
 AnsiString sx,sy,sc;
 int a=s.Pos("+");
 if(a==0)a=s.Pos("-");
 if(a==1){
  sc=s.SubString(2,s.Length()-1);
  a=sc.Pos("-")+1;
 }
 sx=s.SubString(1,a-1);
 sy=s.SubString(a,s.Length()-a);
 x=sx.ToDouble();
 y=sy.ToDouble();
}
AnsiString complex::ToStr(){
 AnsiString s="";
 s=s+AnsiString(x);
 if(y>=0)s=s+"+";
 s=s+AnsiString(y)+"i";
 return s;
}
complex ToComplex(AnsiString s){
 complex z(s);
 return z;
}
complex operator+(complex a,complex b){
 complex c;
 c.x=a.x+b.x;
 c.y=a.y+b.y;
 return c;
}
complex operator-(complex a,complex b){
 complex c;
 c.x=a.x-b.x;
 c.y=a.y-b.y;
 return c;
}
complex operator*(complex a,complex b){
 complex c;
 c.x=a.x*b.x-a.y*b.y;
 c.y=a.x*b.y+a.y*b.x;
 return c;
}
complex operator/(complex a,complex b){
 complex c;
 c.x=(a.x*b.x+a.y*b.y)/(b.x*b.x+b.y*b.y);
 c.y=(-a.x*b.y+a.y*b.x)/(b.x*b.x+b.y*b.y);
 return c;
}
```

Prelucrăm evenimentul *Button1Click*

```
complex a(Edit1->Text);
complex b(Edit2->Text);
complex c=a+b;
AnsiString rez;
```

```
rez="("+a.ToStr()+")"+"+"+"("+b.ToStr()+")"+"="+c.ToStr();
Label3->Caption=rez;
```

Prelucrăm evenimentul *Button2Click*

```
complex a(Edit1->Text);
complex b(Edit2->Text);
complex c=a-b;
AnsiString rez;
rez="("+a.ToStr()+")"+"-"+"("+b.ToStr()+")"+"="+c.ToStr();
Label3->Caption=rez;
```

Prelucrăm evenimentul *Button3Click*

```
complex a(Edit1->Text);
complex b(Edit2->Text);
complex c=a*b;
AnsiString rez;
rez="("+a.ToStr()+")"+"*"+"("+b.ToStr()+")"+"="+c.ToStr();
Label3->Caption=rez;
```

Prelucrăm evenimentul *Button4Click*

```
complex a(Edit1->Text);
complex b(Edit2->Text);
complex c=a/b;
AnsiString rez;
rez="("+a.ToStr()+")"+"/"+"("+b.ToStr()+")"+"="+c.ToStr();
Label3->Caption=rez;
```

Problema7 Evidenţa personalului unei instituţii

Elaboraţi un program prin intermediul căruia se va efectua evidenţa personalului dintr-o instituţie. În instituţia dată sunt următoarele tipuri de personal :

Angajat – despre care se cunosc datele: nume, prenume, codul fiscal, numărul de ore lucrate, plata pentru o oră.

În calitate de metode vor fi: *citirea* datelor de la tastatură, afişarea datelor la ecran, determinarea salariului, conform formulei:

*salariu=ore_lucrate*plata_ora*.

Student – despre care se cunosc datele: nume, prenume, codul fiscal, grupa, media.

În calitate de metode vor fi: *citirea* datelor de la tastatură, afişarea datelor la ecran, determinarea bursei. Bursa studentului o vom determina astfel:

$$Bursa = \begin{cases} 0\ lei, dacă\ \ media < 7.5 \\ 300\ lei, dacă\ \ 7.5 \le media < 8.5 \\ 400\ lei, dacă\ \ 8.5 \le media < 9.5 \\ 500\ lei, dacă\ \ media \ge 9.5 \end{cases}$$

Student Angajat – despre care se cunosc datele: nume, prenume, codul fiscal, grupa, media, numărul de ore lucrate, plata pentru o oră.

În calitate de metode vor fi: *citirea* datelor de la tastatură, afişarea datelor, determinarea bursei şi a salariului. Metodele bursa şi salariu vor efectua calculele

conform formulelor din clasele precedente.

Asupra personalului vor fi efectuate operaţii de adăugare a unei persoane, excludere, afişare a tuturor persoanelor din instituţie, cît şi determinarea bugetului lunar al personalului instituţiei.

Realizare:

Forma aplicaţiei va arăta astfel:

Pe suprafaţa formei plasăm corespunzător componentele:

GroupBox1, modificăm proprietatea *Caption* în *Indicati datele despre personal*. Plasăm în interiorul acesteia componentele:

RadioGroup1 şi efectuăm setările:

Caption=Indicaţi tipul

Items adăugăm*: Student, Angajat, Student Angajat*

Columns=3

8 componente de tip TEdit, modificînd proprietatea *Name* în mod corespunzător: *nume, prenume, idnp, grupa, media, functia, ore, plata*. Pentru fiecare componentă se va modifica proprietatea *Text* în şir vid.

8 componente de tip TLabel;

Button1 – pentru adăugarea informaţiei, setăm proprietatea *Caption=Adauga*;

Pe suprafaţa mai plasăm componentele:

Button2 – pentru afişarea personalului, setăm proprietatea *Caption=Afisare personal*;

Button3 – pentru afişarea sumei ce necesită a fi achitată, setăm proprietatea

Caption=Suma ce necesita a fi achitata;
Button4 – pentru a exclude din personal, setăm proprietatea *Caption=Excludeti*;
Memo1 – pentru afişarea informaţiei.

Codul sursă:

```
class Persoana{
 protected:
 AnsiString nume,prenume,idnp,tip;
 public:
 Persoana(){tip="Persoana";}
 virtual AnsiString afisare();
 AnsiString cod(){return idnp;}
 virtual void citire();
 virtual double salariu(){return 0;}
 virtual double bursa(){return 0;}
};
class Angajat:virtual public Persoana{
 protected:
 AnsiString functia;
 int ore;
 double pl_ora;
 public:
 Angajat(){tip="Angajat";}
 void citire();
 AnsiString afisare();
 double salariu();
};
class Student:virtual public Persoana{
 protected:
 AnsiString grupa;
 double media;
 public:
 Student(){tip="Student";}
 void citire();
 AnsiString afisare();
 double bursa();
};
class Stud_Ang:public Student,public Angajat{
 public:
 Stud_Ang(){tip="Student angajat";}
 void citire();
 AnsiString afisare();
};
void Persoana::citire(){
 nume=Form1->nume->Text;prenume=Form1->prenume->Text;
 idnp=Form1->idnp->Text;}
AnsiString Persoana::afisare(){
 AnsiString s=tip+" "+nume+" "+prenume+" "+idnp+" ";
 return s;}
void Angajat::citire(){
 Persoana::citire();
 functia=Form1->functia->Text;
 ore=Form1->ore->Text.ToInt();
```

```
 pl_ora=Form1->plata->Text.ToDouble();}
AnsiString Angajat::afisare(){
 AnsiString s=Persoana::afisare();
 s=s+functia+" "+FormatFloat("0.00",salariu());
 return s;}
double Angajat::salariu(){return ore*pl_ora;}
void Student::citire(){
 Persoana::citire();
 grupa=Form1->grupa->Text;
 media=Form1->media->Text.ToDouble();}
AnsiString Student::afisare(){
 AnsiString s=Persoana::afisare();
 s=s+grupa+" "+FormatFloat("0.00",bursa());
 return s;}
double Student::bursa(){
 if(media<7.5) return 0;
 else if(media<8.5) return 300;
 else if(media<9.5) return 400;
 else return 500;}
void Stud_Ang::citire(){
 Persoana::citire();
 grupa=Form1->grupa->Text;
 media=Form1->media->Text.ToDouble();
 functia=Form1->functia->Text;
 ore=Form1->ore->Text.ToInt();
 pl_ora=Form1->plata->Text.ToDouble();}
AnsiString Stud_Ang::afisare(){
 AnsiString s=Persoana::afisare();
 s=s+grupa+" "+functia+" "+FormatFloat("0.00",bursa()+salariu());
 return s;}
class celula{
 public:
 Persoana *p;
 celula *next;
 celula(){next=NULL;}
 void cit(int);
 AnsiString afis(){return p->afisare();}
 double consum();
};
double celula::consum(){return p->bursa()+p->salariu();}
void celula::cit(int i){
 switch(i){
  case 0 : p=new Student;break;
  case 1 : p=new Angajat;break;
  case 2 : p=new Stud_Ang;break;
 }p->citire();}
class lista{
 celula *prim;
 public:
 lista(){prim=NULL;}
 void afisare();
 void inserare(int);
 void exclude(AnsiString);
 double bani();
```

```
 ~lista();
};
void lista::afisare(){
 Form1->Memo1->Clear();
 if(prim==NULL) Form1->Memo1->Lines->Add("Lista este vida");
 else{
 Form1->Memo1->Lines->Add("Lista Personalului");
  celula *p;p=prim;
  while(p!=NULL){
   Form1->Memo1->Lines->Add(p->afis());p=p->next;
  }}}
void lista::inserare(int i){
 celula *q;q=new celula;
 q->cit(i);q->next=prim;prim=q;}
void lista::exclude(AnsiString c){
 celula *w,*q;w=prim;
 if(w->p->cod()==c){prim=w->next;delete w;}
 else{while(w->next->p->cod()!=c && w!=NULL)w=w->next;
  if(w!=NULL){q=w->next;w->next=q->next;delete q;}}}
lista::~lista(){
 celula *p,*q;
 if(prim!=NULL)
 do{p=prim;prim=prim->next;delete p;}while(prim!=NULL);}
double lista::bani(){double d=0;celula *p;p=prim;
 while(p!=NULL){d=d+p->consum();p=p->next;}
 return d;}
```

Declaraţii:

```
lista p;
```

Prelucrăm evenimentul *Button1Click*

```
p.inserare(RadioGroup1->ItemIndex);
Button2Click(Sender);
RadioGroup1->ItemIndex=-1;
nume->Clear();prenume->Clear();
idnp->Clear();grupa->Clear();
media->Clear();functia->Clear();
ore->Clear();plata->Clear();
```

Prelucrăm evenimentul *RadioGroup1Click*

```
switch(RadioGroup1->ItemIndex){
 case 0: Button1->Caption="Adauga student";break;
 case 1: Button1->Caption="Adauga angajat";break;
 case 2: Button1->Caption="Adauga student angajat";break;
}
```

Prelucrăm evenimentul *Button2Click*

```
p.afisare();
```

Prelucrăm evenimentul *Button3Click*

```
ShowMessage("Suma ce necesita a fi achitata este "+
FormatFloat("0.00",p.bani()));
```

Prelucrăm evenimentul *Button4Click*

```
AnsiString s=InputBox("Excludere","Indicati codul fiscal al
persoanei ce urmeaza a fi exclusa","");
p.exclude(s);Button2Click(Sender);
```

Probleme pentru rezolvare independentă

1. Forme (ferestre).

1. Elaboraţi o aplicaţie prin intermediul căreia va fi afişată o fereastră cu următoarele setări: dimensiunile – 400×400, culoarea – albastră, poziţia – centru sus, titlu – numele şi prenumele studentului, utilizatorului i se va interzice redimensionarea formei.

2. Elaboraţi o aplicaţie care la efectuarea unui click pe suprafaţa formei va modifica, în mod aleator, culoarea formei (minim 5 culori).

Indicaţie : pentru generarea culorilor va fi utilizată funcţia *random*.

3. Elaboraţi o aplicaţie care la efectuarea unui click va exclude bara de titlu a formei, iar la efectuarea double-click va afişa bara de titlu a formei cu titlul respectiv ”*SALUT*”.

4. Elaboraţi o aplicaţie care va modifica titlul formei în dependenţă de evenimentul care a avut loc. De exemplu, la efectuarea double-click pe suprafaţa formei titlul va fi „A fost efectuat double-click”.

5. Elaboraţi o aplicaţie care la efectuarea unui double-click va modifica consecutiv valoarea proprietăţii BorderStyle.

6. Elaboraţi o aplicaţie care:
a) la apăsarea tastei *x* va închide aplicaţia;
b) la apăsarea tastei *L* va mări lungimea ferestrei cu 10%;
c) la apăsarea tastei *l* va micşora lungimea ferestrei cu 10%;
d) la apăsarea tastei *H* va mări înălțimea ferestrei cu 10%;
e) la apăsarea tastei *h* va micşora înălțimea ferestrei cu 10%;
f) la apăsarea tastei *c* va poziţiona fereastra la centru;
g) la apăsarea tastei + va mări dimensiunea ferestrei cu 10%;
h) la apăsarea tastei - va micşora dimensiunea ferestrei cu 10%.

7. Elaboraţi o aplicaţie care :
a) la apăsarea tastei ↑ va deplasa fereastra în sus;
b) la apăsarea tastei ↓ va deplasa fereastra în jos;
c) la apăsarea tastei → va deplasa fereastra în dreapta;
d) la apăsarea tastei ← va deplasa fereastra în stînga.

8. Elaboraţi o aplicaţie care va afişa iniţial forma la centrul suprafeţei de lucru. La apăsare tastelor ↑,↓ forma se va deplasa pe diagonala principală a suprafeţei de lucru. Apăsarea caracterului '*s*' va poziţiona forma în centru, iar la apăsarea tastelor ↑,↓ forma se va deplasa pe diagonala secundară. Apăsarea caracterului '*p*' va poziţiona forma în centru, iar forma se va deplasa pe diagonala principală la apăsarea tastelor ↑,↓.

9. Elaboraţi o aplicaţie care va afişa forma pe centru suprafeţei de lucru cu dimensiunile 300×300. Se va interzice modificare dimensiunilor formei. La efectuarea unui click pe suprafaţa formei aceasta se va deplasa consecutiv: stînga sus, dreapta sus, dreapta jos, stînga jos, centrul.

10. Elaboraţi o aplicaţie care va permite redimensionarea formei de la tastatură. Pentru a redimensiona înălţimea utilizatorul va tasta caracterul '*h*', apoi o cifră *c* din intervalul [1,8], iar în dependenţă de cifra tastată înălţimea formei va fi 100*c. Pentru modificarea lăţimii se va proceda în mod similar prin tastarea caracterului '*w*'.

2. Componenta Button.

1. Creaţi aplicaţia SHOW/HIDE. Pe suprafaţa formei cu titlul "SHOW/HIDE" sunt plasate 3 butoane, cu titlurile respective: show, hide, show/hide. La efectuarea unui click pe unul din primele 2 butoane butonul show/hide va fi ascuns sau afişat, în dependenţă de butonul apăsat.

2. Creaţi aplicaţia MOUSE. Pe suprafaţa formei cu titlul "MOUSE" sunt plasate 3 butoane, cu titlurile respective: crCros, crHelp, crNoDrop. La efectuarea unui click pe unul dintre butoane, pe suprafaţa formei mausul va lua forma cu proprietatea corespunzătoare titlului butonului.

3. Creaţi aplicaţia *DEPLASARE*. Pe suprafaţa formei cu titlul "*DEPLASARE*" vor fi plasate 4 butoane, cu titlurile respective: sus, jos, dreapta, stînga. Efectuarea unui click pe unul din butoane va deplasa forma cu 20% faţă de poziţia precedentă a formei.

4. Creaţi aplicaţia *MARESTE*. Pe suprafaţa formei cu titlul "*MARESTE*" va fi plasat un singur buton cu titlul "+", butonul va fi poziţionat în mijlocul formei. Efectuarea unui click pe suprafaţa butonului va mări dimensiunea acestuia cu 15%, dacă dimensiunea butonului depăşeşte dimensiunea formei, atunci forma se închide.

5. Creaţi aplicaţia *OPERATII.* Pe suprafaţa formei cu titlul ”*OPERATII*” vor fi plasate 2 butoane. Primul buton va fi implicit (proprietatea Default setată la *true)* efectuarea unui click pe suprafaţa acestuia sau apăsarea tastei Enter va modifica titlul acestuia în *”AM TASTAT ENTER”*, butonul doi cu titlul ”*ESC*”, buton de anulare a formei (proprietatea Cancel setată la *true),* efectuarea unui click pe suprafaţa acestuia sau tastarea tastei ESC va închide forma.

6. Creaţi aplicaţia INCREMENTARE/DECREMENTARE. Pe suprafaţa formei cu titlul ”*0*” se plasează două butoane cu titlurile respectiv ”*incrementare*”/”*decrementare*”. Efectuarea unui click pe unul dintre butoane va modifica titlul formei incrementînd o unitate, respectiv decrementînd o unitate.

7. Creaţi aplicaţia *DEPLASARE*. Pe suprafaţa formei cu titlul ”*DEPLASARE*” se plasează un buton fără titlu. La tastarea tastelor ↑, ↓, →, ← se va deplasa butonul, în dependenţă de tasta apăsată, cu 15% faţă de poziţia precedentă. Dacă butonul „iese” de pe suprafaţa formei, atunci el revine în partea opusă.

8. Creaţi aplicaţia CULOARE. Pe suprafaţa formei cu titlul ”CULOARE” vor fi plasate 5 butoane cu titlurile respectiv: rosu, verde, galben, negru, alb. Efectuarea unui click pe unul din butoane va modifica coloare formei corespunzător butonului apăsat.

9. Creaţi aplicaţia POZITIE. Pe suprafaţa formei cu titlul ”POZITIE” vor fi plasate 6 butoane cu titlurile respectiv: stînga sus, dreapta sus, dreapta jos, stînga jos, centru, deplasare. La efectuarea unui click pe unul din primele 5 butoane, butonul cu titlul ”deplasare”, se va deplasa în concordanţă cu titlul butonului apăsat.

10. Creaţi aplicaţia SCARA. Pe suprafaţa formei cu titlul ” SCARA” se plasează 5 butoane. La efectuarea unui click pe suprafaţa formei butoanele vor avea dimensiunile 30*80, titlurile corespunzător 1, 2, 3, 4, 5 şi vor fi aranjate în formă de scară (de la 1 la 5).

3. Componenta Edit

1. Creaţi aplicaţia TITLUL FORMEI. Pe suprafaţa formei sunt plasate componentele Buton (cu titlul ”*Scrie*”) şi Edit. La efectuarea unui click pe suprafaţa butonului titlul ferestrei va corespunde cu textul din cutia Edit.

2. Creaţi aplicaţia COPY_PASTE. Pe suprafaţa formei sunt plasate două butoane şi două cutii de editare. La efectuarea unui click pe primul buton se va copia textul din prima cutie în cutia a doua, iar la efectuarea unui click pe butonul doi se va copia textul din a doua cutie în prima. Modificaţi titlul butoanelor în conformitate cu operaţiile care se efectuează. La elaborarea acestei aplicaţii se vor utiliza metodele caracteristice pentru prelucrarea textului din cutie (*CopyToClipboard*, *PasteFromClipboard*).

3. Creaţi aplicaţia SCRIE. Pe suprafaţa formei sunt plasate două butoane cu titlurile ”*Scrie*” şi ”*Interzice*”. După efectuare unui click pe butonul *Scrie* utilizatorul va avea posibilitatea de a edita text în cutie. Efectuarea unui click pe butonul *Interzice* nu va permite utilizatorului editarea textului în cutie.

4. Creaţi aplicaţia CARACTER. Pe suprafaţa formei sunt plasate trei cutii de editare. În fiecare dintre cutii la întîlnirea caracterului *a*, acesta va fi exclus, dublat, respectiv triplat.

5. Creaţi aplicaţia EDITARE. Pe suprafaţa formei sunt două cutii de editare. În prima cutie se va permite numai editarea cifrelor, în cutia a doua numai caractere ale alfabetului latin.

6. Creaţi aplicaţia TRANSFORMA. Pe suprafaţa formei cu titlul ”TRANSFORMA” se plasează 3 butoane cu titlurile respectiv: *minuscule*, *majuscule*, *normal* şi o componentă Edit. În cutia de editare utilizatorul va culege un text. La efectuarea unui click pe butonul majuscule, textul din cutie va transformat în majuscule, similar minuscule, iar la efectuarea unui click pe butonul *normal*, textul va reveni la forma iniţială.

7. Creaţi aplicaţia LOGARE. Pe suprafaţa formei cu titlul ”LOGARE” se plasează 3 componente Edit şi un buton implicit cu titlul *Validare*. În prima componentă se va scrie numele utilizatorului, în a doua componentă parola (textul pentru parolă va fi înlocuit cu caracterul ’*’). La efectuarea unui click pe butonul Validare, a treia cutie de editare va conţine un mesaj care va specifica succesul sau insuccesul logării. În cazul în care logarea a fost cu succes, numele utilizatorului va fi scris în titlul ferestrei.

8. Creaţi aplicaţia VOCALE. Pe suprafaţa formei este plasată componenta *Edit1*. La introducerea textului în cutie, titlul formei va indica numărul de apariţii ale vocalelor în cutie.

9. Creaţi aplicaţia COLOANE. Pe suprafaţa formei cu titlul ”COLOANE” se plasează 4 componente Edit şi 4 componente Buton. La efectuarea unui click pe suprafaţa formei componentele Buton vor fi aranjate într-o coloană, iar componentele Edit într-o altă coloană.

10. Creaţi aplicaţia OnKeyPress. Pe suprafaţa formei este plasată o componentă Edit. De prelucrat evenimentul OnKeyPress al componentei astfel încît:

a) la tastarea caracterului ‘q’ tot textul din cutie se va dubla;
b) la tastarea caracterului ‘w’ tot textul din cutie se va tripla;
c) la tastarea caracterului ‘e’ textul din cutie va fi exclus;
d) la tastarea caracterului ‘r’ titlul formei va fi identic cu textul din cutie.

4. Şiruri de caractere şi componenta Label

1. Creaţi aplicaţia *Sir_1*. Pe suprafaţa formei cu titlul ”Sir_1” se plasează 2 componente Edit, un buton cu titlul *Caută* şi o componentă Label. În prima componentă Edit se va scrie o frază, în a doua un cuvînt (silabă, caracter). La efectuarea unui click pe butonul *Caută*, în componenta Label se va afişa poziţia iniţială în şir a cuvîntului căutat. În cazul în care cuvîntul nu se include în frază, va fi afişat un mesaj corespunzător.

2. Creaţi aplicaţia *Sir_2*. Pe suprafaţa formei cu titlul ”Sir_2” se plasează 2 componente Edit, un buton cu titlul *Exclude* şi o componentă Label. În prima componentă Edit se va scrie o frază, în a doua un cuvînt (silabă, caracter). La efectuarea unui click pe butonul *Exclude*, cuvîntul din fraza editată va fi exclus din frază, rezultatul va fi afişat prin intermediul componentei Label.

3. Creaţi aplicaţia *Sir_3*. Pe suprafaţa formei cu titlul ”Sir_3” se plasează 2 componente Edit, un buton cu titlul *Numara* şi o componentă Label. În prima componentă Edit se va scrie o frază, în a doua un cuvînt (silabă, caracter). La efectuarea unui click pe butonul *Numara*, în componenta Label se va scrie numărul de apariţii a cuvîntului în frază.

4. Creaţi aplicaţia *Replace*. Pe suprafaţa formei cu titlul ”Replace” se plasează 3 componente Edit, un buton cu titlul R*eplace* şi o componentă Label. În prima componentă Edit se va scrie o frază, în a doua un cuvînt (silabă, caracter) care urmează a fi înlocuit, în a treia componenta se va scrie textul cu care urmează a fi înlocuit. La efectuarea unui click pe butonul *Replace*, va avea loc înlocuirea textului. Rezultatul va fi scris în componenta Label.

5. Creaţi aplicaţia *Palindrom*. Într-o cutie de editare se intoduce un cuvînt. La efectuarea unui click pe butonul *Verifica* se va afişa prin intermediul unei componente Label, dacă cuvîntul introdus este palindrom.

6. Creaţi aplicaţia *Cifre*. Pe suprafaţa formei cu titlul ”Cifre” se va plasa o componentă Edit, care va permite numai editarea cifrelor. La efectuarea unui click pe butonul *Numara*, în componenta label va fi afişată cifra care apare de cele mai multe ori, de cele mai puţine ori, cît şi numărul de repetări ale fiecărei cifre.

7. Creaţi aplicaţia *Inversul*. Pe forma cu titlul ”Inversul” sunt plasate componentele Edit, Buton cu titlul *Modifica* şi o componentă Label. În cutia de editare se vor scrie mai multe cuvinte. La efectuarea unui click pe butonul *Modifica*, fiecare cuvint va fi înlocuit cu inversul său (de la sfîrşit la început). Şirul modificat va fi afişat în componenta Label.

8. Creaţi aplicaţia *Cuvînt maxim*. Pe suprafaţa formei cu titlul ”Cuvînt maxim” sunt plasate componentele Edit, Buton cu titlul *Determina* şi o componentă Label. În cutia de editare se vor scrie mai multe cuvinte. La efectuarea unui click pe butonul *Determina*, în componenta Label va fi scris cel mai mare cuvînt.

9. Creaţi aplicaţia *Litere*. Pe suprafaţa formei cu titlul ”Litere” se va plasa o componentă Edit, în care se va scrie un şir de caractere. La efectuarea unui click pe butonul *Exclude*, în componenta label va fi afişat şirul fără de cifre.

10. Creaţi aplicaţia *ASCII*. Pe forma cu titlul ”ASCII” sunt plasate componentele Edit, Buton cu titlul *Arata* şi o componentă Label. În cutia de editare se vor scrie caractere. La efectuarea unui click pe butonul Arata în componenta Label vor fi scrise codurile ASCII ale caracterelor scrise în componenta Edit.

11. Creaţi aplicaţia *Calculator*. Prin intermediul a trei cutii de editare utilizatorul introduce două valori întregi şi un operator (+, -, *, /, %). La efectuarea unui click pe butonul *Calcul* se va afişa prin intermediul unei componente Label rezultatul operaţiei dintre cele două valori şi operatorul indicat.

12. Elaboraţi o aplicaţie care va verifica dacă greutatea şi înălţimea unui copil corespund vîrstei. Vîrsta copilului se va introduce prin intermediul unei componente Edit. Formule de calcul :

greutate=2*v+8 (în kg),

înălţime=5*v+80 (în cm).

La efectuarea unui click pe butonul *Determina* se va indica în 2 componente Label greutatea şi înălţimea determinate după formulele de mai sus.

13. Elaboraţi o aplicaţie care să modeleze jocul *Ghici numarul*. Jocul constă în faptul că este generat un număr aleator pe intervalul 0..9 de către calculator. Utilizatorul întroduce o valoare de la 0 la 9 într-o cutie de editare, după efectuarea unui click pe butonul GHICI, este afişată prin intermediul unei componente Label valoarea dată de calculator. În cazul în care utilizatorul a ghicit, va fi afişat un mesaj care va indica acest fapt.

14. Creaţi aplicaţia *Ecuaţii*, care va permite determinarea soluţiilor următoarelor ecuaţii :

a) $ax+b=0$;

b) $ax^2 + bx + c = 0$.

Rezultatele vor fi afişate prin intermediul componentelor Label.

15. Creaţi aplicaţia *Sistem de ecuaţii*, care să determine soluţiile unui sistem de ecuaţii cu două necunoscute. Rezultatele vor fi afişate prin intermediul componentelor Label.

16. Creaţi aplicaţia *Triunghi*. Prin intermediul a 3 cutii de editare se introduc 3 valori numerice, care reprezintă lungimile laturilor unui triunghi. La efectuarea unui click pe butonul *Calcul* se va determina:

a) tipul triunghiului (ascuţit, obtuz, dreptunghic), utilizînd teorema lui Pitagora;

b) aria triunghiului (Formula lui Heron);

c) perimetrul triunghiului.

În cazul în care valorile introduse nu pot forma un triunghi, va fi afişat un mesaj, care va indica aceasta şi nu se va efectua nici un calcul. Rezultatele vor fi afişate prin intermediul componentelor Label.

17. Creaţi aplicaţia *Divizor şi Multiplu comun*. Prin intermediul a două componente Edit se citesc două numere întregi. La efectuarea unui click pe butonul *Determina* în componenta Label se va fişa cel mai mare divizor comun, cît şi cel mai mic multiplu comun.

18. Creaţi aplicaţia *Numere prime*. Prin intermediul unei componente Edit se citeşte un număr întreg. La efectuarea unui click pe butonul *Determina* într-o componentă Label vor fi afişate toate numerele prime mai mici sau egale cu numărul

citit.

19. Creaţi aplicaţia *Combinări*. Prin intermediul a două componente Edit se vor citi 2 numere întregi *n* şi *r*. Într-o componentă Label se va afişa valoarea: C_n^r, calculată conform formulei:

$$C_n^r = \frac{n!}{r!*(n-r)!}$$

20. Creaţi aplicaţia *Numere prietene*. Prin intermediul a două componente Edit se citesc 2 numere întregi. Două numere se numesc prietene dacă posedă proprietatea: fiecare din ele este egal cu suma divizorilor inferiori celuilalt, de exemplu 220 şi 284. Prin intermediul unei componente Label se va afişa dacă numerele sunt sau nu prietene.

21. Creaţi aplicaţia *Numar perfect*. Prin intermediul unei componente Edit se citeşte un număr întreg. Prin intermediul unei componente Label se va afişa dacă numărul este perfect, supraperfect sau imperfect. Numărul este perfect dacă suma divizorilor inferiori acestuia este egala cu el însuşi, dacă suma este mai mică ca numărul atunci el este imperfect, dacă suma este mai mare ca numărul dat atunci el se numeşte supraperfect.

5. Aplicaţii multiforme.

1. Elaboraţi o aplicaţie multiformă. Aplicaţia va fi formată din 3 forme. Pe suprafaţa formei principale (forma 1) este plasat un buton cu titlul *Forma Noua*. La efectuarea unui click pe suprafaţa acestuia, este ascunsă forma principală şi afişată o altă formă similară formei principale, titlul acestei forme fiind *Forma 1*. Efectuarea unui click pe butonul *Forma Noua* ascunde forma şi afişează o altă forma cu titlul *Forma 2*, ş.d. Închiderea unei dintre forme va genera închiderea aplicaţiei.

2. Elaboraţi o aplicaţie care la efectuarea unui click pe butonul cu titlul *Intro*, va afişa o fereastră de dialog prin care i se va cere utilizatorului să-şi scrie prenumele (de exemplu Ion). După efectuarea unui click pe butonul *Ok*, fereastra va fi închisă şi va fi afişată o alta fereastră, cu un mesaj de salut (de exemplu Salut Ion).

3. Elaboraţi o aplicaţie prin intermediul căreia la efectuarea unui click pe butonul *Afisare* de pe suprafaţa formei, vor fi afişate 8 ferestre de dialog aranjate pe diagonala principală a monitorului.

4. Creaţi aplicaţia *Tabla Inmultirii*. Pe suprafaţa formei cu titlul ”Tabla Inmultirii”, este plasat butonul *Verifica*. Efectuarea unui click pe suprafaţa acestuia, afişează o fereastră de dialog. Această fereastră conţine o întrebare de genul *x*y= ?*, unde *1<x,y<11*. După scrierea răspunsului corect de către utilizator este apăsat butonul Ok. Dacă răspunsul este corect fereastra este închisă, în caz contrar, fereastra nu este închisă pînă cînd nu este introdus răspunsul corect. La realizarea aplicaţiei vor fi utilizate forme modale.

5. Creaţi aplicaţia *Virus*. Pe suprafaţa formei principale se plasează un buton cu titlul *Lansează virus*. La efectuarea unui click pe acest buton forma principală este ascunsă şi afişată o altă formă cu titlul *Virus*. Închiderea formei cu titlul *Virus*, va afişa o nouă formă cu acelaşi titlu. Aplicaţia se va închide la apăsarea tastei *x*.

6. Elaboraţi o aplicaţie multiformă. Prin intermediul formei principale se vor seta unele proprietăţi ale formei a doua: titlul, dimensiunile, distanţa de sus, distanţa din stînga. Numai după introducerea acestor date, la efectuarea unui click pe butonul *Arata*, va fi afişată a doua formă, iar forma principală va fi ascunsă. La închiderea formei secundare, forma principală va fi afişată.

7. Creaţi aplicaţia *Bursa*. Prin intermediul formei principale utilizatorul va introduce datele despre un student: numele, grupa, anul naşterii şi media notelor curente. La efectuarea unui click pe butonul *determină media*, va fi afişată o fereastră cu titlul: numele şi grupa studentului, în interiorul acesteia va fi afişată vîrsta şi media studentului. Pentru determinarea bursei va fi utilizată formula: *media notelor curente**75, dacă media notelor curente este mai mare ca 7.6, în caz contrar 0.

8. Creaţi aplicaţia *Dialog*. Pe suprafaţa formei principale vor fi plasate 2 butoane de comandă, cu titlurile respectiv, *Inchide aplicaţia*, *Modifica culoarea formei*. La efectuarea unui click pe unul dintre butoane, va fi afişată o fereastră de dialog cu următorul conţinut:

Inchide aplicaţia – fereastra de dialog va conţine butoanele *Yes* şi *No*. La efectuarea unui click pe butonul *Yes*, aplicaţia se va închide;

Modifica culoarea formei – fereastra conţine un cîmp pentru introducerea culorii şi butoanele *Ok*, *Cancel*. Efectuarea unui click pe butonul *Ok*, modifică culoarea formei corespunzător valorii introduse.

9. Creaţi aplicaţia *Test*. Pe suprafaţa ferestrei principale se plasează un buton de comandă cu titlul *Test*. La efectuarea unui click pe suprafaţa acestuia este afişată o formă modală. Pe suprafaţa acestei forme este plasată o componentă Label (în

interiorul acesteia este afişat un cod aleator format din 4 cifre), o componentă Edit (în cadrul acestei componente utilizatorul va scrie codul afişat în Label), un buton de comandă. La efectuarea unui click pe butonul de comandă se va verifica corectitudinea codului, dacă codul este scris corect, atunci fereastra modală este închisă. La elaborarea aplicaţiei vor fi excluse alte posibilităţi de închidere a ferestrei modale.

10. Elaboraţi o aplicaţie care la lansare va afişa o formă, prin intermediul căreia se va cere utilizatorului să introducă numele şi parola. Validarea acestora se va efectua prin efectuare unui click pe butonul *Validare* sau apăsînd tasta *Enter*. Dacă numele şi parola sunt corecte se va deschide o nouă formă, iar precedenta va fi ascunsă. Scrierea greşită a numelui sau a parolei de trei ori consecutiv va închide forma proiectului.

6. Aplicaţii cu butoane

1. Creaţi aplicaţia CONTOR. Pe suprafaţa ferestrei principale sunt plasate componentele: Edit, un buton UpDown asociat componentei Edit, un grup de 4 butoane radio, cu titlurile respectiv unităţi, zeci, sute, mii. La efectuarea unui click pe unul dintre butoanele radio valoarea din cutia de editare va fi incrementată sau decrementată corespunzător butonului radio.

2. Creaţi aplicaţia GRUPURI. Pe suprafaţa ferestrei principale sunt plasate două grupuri de butoane radio, o cutie de editare şi un buton de comandă. Primul grup conţine 3 opţiuni (forma, cutia, ambele). Al doilea grup conţine 6 opţiuni (diferite culori). La efectuarea unui click pe butonul de comandă pentru formă şi pentru cutia de editare vor fi efectuate modificările setate corespunzător selecţiei efectuate. De ex. dacă în primul grup este bifată opţiunea *ambele*, iar al în al doilea opţiunea *galben*, atunci atît forma cît şi cutia vor avea culoarea galbenă.

3. Elaboraţi o aplicaţie care să modeleze jocul *Răspuns corect.* Utilizatorului i se adresează un set de 5 întrebări cu 4 variante de răspuns. Utilizatorul va bifa răspunsul, pe care îl consideră corect, după care va efectua un click pe butonul cu titlul *Raspuns*. Efectuarea operaţiei click va afişa un mesaj prin care se va indica numărul de răspunsuri corecte.

4. Creaţi aplicaţia POZIŢIE. Pe suprafaţa ferestrei principale sunt plasate 3 grupuri de butoane radio:

a) Grupul *Pozitie* va conţine următoarele opţiuni: *Centru, Stinga sus, Stinga jos, Dreapta sus, Dreapta jos*.
b) Grupul *Culoare* va conţine o listă cu 5 opţiuni cu diferite culori.
c) Grupul *Dimensiune* va conţine următoarele opţiuni: *200×200, 200×300, 300×200, 400×500, 500×400*.

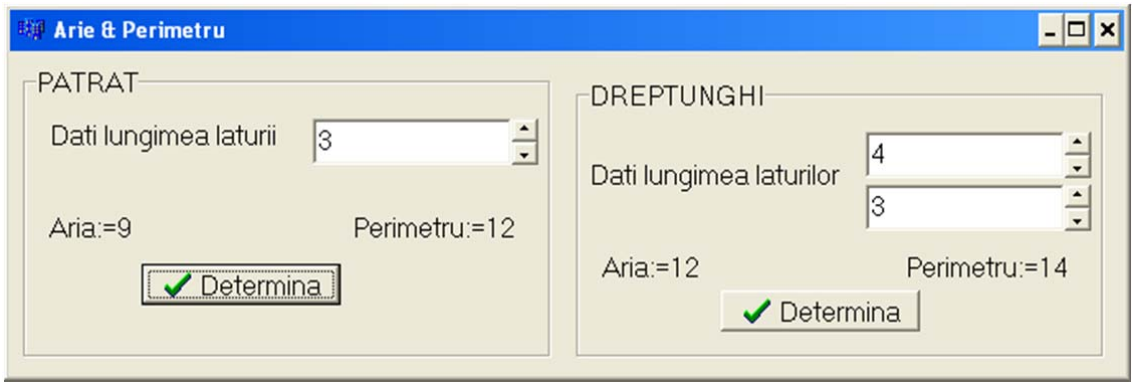
Efectuarea unui click pe fiecare grup va afişa forma cu setările selectate.

5. Creaţi aplicaţia CONVERTOR. Această aplicaţie va permite efectuarea următoarelor transformări:

a) milimetri, centimetri, decimetri, kilometri, dacă valoarea iniţială se consideră dată în metri;
b) miligrame, kilograme, tone, centigrame, dacă valoarea iniţială se consideră dată în grame;
c) săptămîni, zile, minute, secunde, dacă valoarea iniţială se consideră dată în ore.

La elaborarea aplicaţiei vor fi utilizate componente RadioGroup (cîte un grup pentru fiecare caz). Pentru fiecare caz va fi afişat rezultatul în componente Label.

6. Creaţi aplicaţia *Arie & Perimetru*. Forma aplicaţiei va arăta astfel:



La efectuarea unui click pe butonul *Determina* va fi afişată aria şi perimetrul figurii corespunzătoare. Modificarea lungimilor laturilor va modifica în mod corespunzător aria şi perimetru figurilor.

7. Creaţi aplicaţia *Triunghi* în baza aplicaţiei *Arie & Perimetru*.

8. Creaţi aplicaţia *Test*. Utilizatorului i se adresează o întrebare şi 8 variante de răspuns. Răspunsurile vor fi afişate prin intermediul unui grup de butoane radio. Prin intermediul unei componente Edit şi al unui buton de tip TUpDown, utilizatorul va avea posibilitatea de a micşora/mări numărul variantelor de răspuns pe segmentul [3;8]. La bifarea răspunsului va fi afişat un mesaj în care se va indica corectitudinea răspunsului.

9. Creaţi aplicaţia *Baza*. Pe suprafaţa formei sunt plasate 8 componente CheckBox, 8 componente Label, o cutie de editare. Prin intermediul cutiei de editare se va introduce un număr întreg. Butoanele de tip TCheckBox vor avea titlurile: *baza doi*, *baza trei*,..., *baza nouă*. La bifarea butoanelor numărul introdus va fi transformat corespunzător bazei selectate şi afişat (doar dacă este bifat) prin intermediul componentelor de tip TLabel.

10. Creaţi aplicaţia *Butoane Radio*. Pe suprafaţa formei cu titlul ”Butoane Radio” este plasată o componentă *RadioGroup* şi un buton de comandă. La efectuarea unui click pe suprafaţa butonului de comandă în componenta RadioGroup vor fi adăugate 10 butoane cu titlurile respectiv 0,1,...,9. Butoanele vor fi divizate în trei coloane. La selectarea butonului cu valoare maximă, în cutie va fi adăugat încă un buton (următorul după numărul maxim). Se va permite adăugarea de noi butoane, doar dacă valoarea butonului maxim nu depăşeşte 50. La selectarea butonului cu titlul 0, va fi exclus butonul cu valoarea maximală, excluderea se va efectua doar dacă numărul de butoane depăşeşte valoare 3. La efectuarea unui click pe oricare alt buton din cutie, va fi afişat un mesaj care va conţine titlul butonului selectat.

7. Aplicaţii cu meniuri

1. Creaţi aplicaţia *Meniu*. Se va crea un meniu principal cu următoarele opţiuni:
   a) *poziţie* cu opţiunile: sus, jos, stînga, dreapta centru;
   b) dimensiuni cu opţiunile: *200×200, 200×300, 300×200, 400×500, 500×400;*
   c) *culoare* cu opţiunile şase tipuri de culori.

La efectuarea unui click pe una dintre opţiunile meniului, structura formei va fi modificată în mod corespunzător.

2. Pentru o formă vor fi creat un meniu contextual cu următoarele opţiuni:
   a) *poziţie* cu opţiunile: sus, jos, stînga, dreapta centru;
   b) *dimensiuni* cu opţiunile: *200×200, 200×300, 300×200, 400×500, 500×400;*
   c) *culoare* cu opţiunile şase tipuri de culori.

La efectuarea unui click pe una dintre opţiunile meniului, structura formei va fi modificată în mod corespunzător.

3. Creaţi aplicaţia *Sir*. Pe suprafaţa formei cu titlul ”Sir” se plasează o componentă Memo şi se creează un meniu principal cu opţiunile:
   a) Cauta – va căuta un text;

b) Exclude – va exclude un text;
c) Inlocuieste – va înlocui un text cu alt text.

4. Creaţi aplicaţia *Caractere*. Pe suprafaţa formei cu titlul ”Caractere” se plasează o componentă Memo şi se creează un meniu contextual, ataşat componentei Memo, cu opţiunile:

a) Vocale – va determina numărul de vocale;
b) Linia maxima – va determina linia de lungime maximă;
c) Linia minima – va determina linia de lungime minimă;
d) Majuscule – va transforma textul în majuscule;
e) Minuscule – va transforma textul în minuscule;
f) Normal – va afişa textul cu formatarea sa iniţială;
g) Save – va salva textul în fişierul date.txt.

5. Creaţi aplicaţia *Număr*. Prin intermediul unei componente de tip TEdit se citeşte un număr format din mai multe cifre (va fi permisă numai editarea cifrelor). La efectuarea unui click pe butonul cu titlul *Proprietati* în componenta Memo se vor afişa:

a) numărul de cifre;
b) prima cifră;
c) ultima cifră;
d) suma cifrelor;
e) scrie numai cifrele pare;
f) scrie numai cifrele impare;

Pe suprafaţa formei va mai fi plasat un buton cu titlul *Save*, acţionarea căruia va permite salvarea informaţiei în fişier.

6. Creaţi aplicaţia *Numere*. Prin intermediul a trei componente de tip TEdit se citesc 3 numere întregi. La efectuarea unui click pe butonul cu titlul *Numere* în componenta Memo se va afişa:

a) divizorii primului număr şi care nu sunt divizori pentru al doilea număr;
b) cel mai mare divizor comun;
c) cel mai mic multiplu comun.

Pe suprafaţa formei va mai fi plasat un buton cu titlul *Save*, acţionarea căruia va permite salvarea informaţiei în fişier.

7. Creaţi aplicaţia *Coduri ASCII*. La elaborarea aplicaţiei va fi utilizată componenta Memo şi va fi creat un meniu contextual cu opţiunile:

ASCII	***Caractere***
Deschide	*Deschide*
Transforma	*Transforma*
Salvare	*Salvare*

Efectuarea unui click pe opţiunea *Deschide* va scrie caracterele din fişier (*ascii.txt/caractere.txt*) în componenta Memo;

Efectuarea unui click pe opţiunea *Transforma* va transforma textul în cod ASCII (va transforma codul în text) , apoi acesta va fi scris în componenta Memo;

Efectuarea unui click pe opţiunea *Salvare* va scrie caracterele din Memo în unul din fişierele (*ascii.txt*, dacă textul a fost transformat în cod şi respectiv *caractere.txt*);

8. Creaţi aplicaţia *Cifre*. La elaborarea aplicaţiei va fi utilizată componenta Memo şi va fi creat un meniu contextual cu opţiunile:

Cifra	***Caractere***
Deschide	*Deschide*
Transforma	*Transforma*
Salvare	*Salvare*

Efectuarea unui click pe opţiunea *Deschide* va scrie textul din fişier (*cifra.txt/caractere.txt*) în componenta Memo;

Efectuarea unui click pe opţiunea *Transforma* va scrie cifrele cu caractere (va transforma textul în cifre) în componenta Memo;

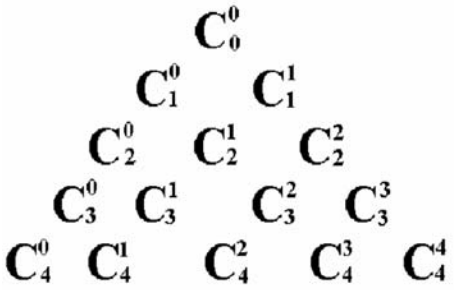
Efectuarea unui click pe opţiunea *Salvare* va scrie caracterele din Memo în unul din fişierele (*caractere.txt*, dacă cifrele sunt scrise cu caractere şi respectiv *cifra.txt*);

9. Creaţi aplicaţia *Fişiere*. Prin intermediul aplicaţiei *Fişiere* se va crea fişierul *Unit.txt* (salvat în directorul curent), care va conţine informaţia din fişierele *Unit.h* şi *Unit1.cpp*. Conţinutul acestui fişier va fi afişat prin intermediul unei componente Memo.

10. Creaţi aplicaţia ”*Triunghiul lui Pascal*”. Prin intermediul unei componente de tip TEdit se va citi înălţimea triunghiului. Triunghiul va fi afişat prin intermediul unei componentei Memo. De exemplu triunghiul cu înălţimea 4 va arăta astfel:

Triunghiul lui Pascal :

$$C_n^r = \frac{n!}{r!*(n-r)!}$$



Pe suprafaţa formei va mai fi plasat un buton cu titlul *Save*, acţionarea căruia va permite salvarea informaţiei în fişier.

8. Aplicaţii cu elemente de grafică

1. Creaţi aplicaţia *Triunghi*. La elaborarea aplicaţiei va fi creat un meniu principal cu opţiunile:

Isoscel	***Oarecare***	***Echilateral***
Ascutit	*Ascutit*	
Dreptunghic	*Dreptunghic*	
Obtuz	*Obtuz*	

Efectuarea unui click pe una dintre opţiunile meniului va desena triunghiul selectat pe suprafaţa unei componente de tip TImage.

2. Creaţi aplicaţia *Patrulater*. La elaborarea aplicaţiei va fi creat un meniu principal cu opţiunile:

Paralelogram	***Trapez***
Oarecare	*Oarecare*
Dreptunghi	*Isoscel*
Patrat	*Dreptunghic*
Romb	

Efectuarea unui click pe una dintre opţiunile meniului va desena patrulaterul selectat pe suprafaţa unei componente de tip TImage.

3. Creaţi aplicaţia *Dreptunghiuri*. Prin intermediul unei cutii de editare se citeşte numărul de dreptunghiuri *n* (1<n<11). La efectuarea unui click pe butonul *Deseneaza*, pe suprafaţa unei componente de tip TImage se desenează *n* dreptunghiuri. Fiecare dreptunghi va avea o culoare diferită de precedentul, lungimile laturilor vor fi cu 10% mai mari ca laturile precedentului dreptunghi şi toate dreptunghiurile vor avea acelaşi centru.

4. Creaţi aplicaţia *Cercuri*. Prin intermediul unei cutii de editare se citeşte numărul de cercuri *n* (1<n<11). La efectuarea unui click pe butonul *Deseneaza*, pe suprafaţa unei componente de tip TImage se desenează *n* cercuri. Fiecare cerc va avea o culoare diferită de precedentul, lungimea razei va fi cu 10% mai mare ca lungimea razei cercului precedent toate cercurile vor avea acelaşi centru.

5. Creaţi aplicaţia *Tabla de şah*. Pe suprafaţa formei se plasează un buton de comandă cu titlul *Deseneaza* şi o componentă de tip TImage. La efectuarea unui click pe butonul cu titlul *Deseneaza*, va fi desenată o tablă de şah pe suprafaţa componentei de tip TImage.

6. Creaţi aplicaţia *Stea*. Pe suprafaţa formei se plasează un buton de comandă cu titlul *Deseneaza* şi o componentă de tip TImage. La efectuarea unui click pe butonul cu titlul *Deseneaza*, va fi desenată o stea pe suprafaţa componentei de tip TImage.

7. Creaţi aplicaţia *Drapel*. La elaborarea aplicaţiei va fi creat un meniu contextual cu numele a cinci ţări. Efectuarea unui click pe una dintre opţiunile meniului va desena drapelul ţării selectate, iar la mijlocul desenului va fi scris numele ţării respective. Pentru elaborarea desenului se va utiliza o componentă de tip TImage.

8. Creaţi aplicaţia *Figuri*. La elaborarea aplicaţiei va fi creat un meniu contextual cu opţiunile: cerc, patrat, dreptunghi. Prin intermediul acestui meniu se va selecta figura ce urmează a fi desenată. La efectuarea unui click pe suprafaţa formei se citesc coordonatele centrului figurii şi se desenează figura selectată din meniu. Efectuarea operaţiei click de mai multe ori, va desena mai multe figuri pe suprafaţa formei.

9. Creaţi aplicaţia *Casa Mea*. Prin intermediul acestei aplicaţii se va desena o casă pe o componentă de tip TImage. Prin intermediul unui buton cu titlul salvează, imaginea va fi salvată în fişierul *casa.jpg*;

10. Creaţi aplicaţia *Figuri geometrice*. La elaborarea aplicaţiei va fi creat un meniu contextual cu opţiunile: triunghi, patrulater. După efectuarea unui click pe una dintre opţiunile meniului, utilizatorul va efectua operaţia click de 3 ori, respectiv 4 ori (se vor citi colţurile figurilor), pe suprafaţa formei. La efectuarea unui click dreapta pe suprafaţa formei va fi afişat un meniu cu opţiunea desen. La efectuarea unui click pe acesta va fi desenat un triunghi sau patrulater.

11. Creaţi aplicaţia *Parabola*. Prin intermediul a trei cutii de editare se vor introduce coeficienţii pentru o ecuaţie de gradul doi (parabolă). La efectuarea unui click pe butonul cu titlul *Desen* va fi desenat pe o componentă de tip TImage graficul ecuaţiei introduse.

12. Creaţi aplicaţia *Hiperbola*. Prin intermediul a patru cutii de editare se vor introduce coeficienţii pentru o ecuaţie de gradul trei (hiperbolă). La efectuarea unui click pe butonul cu titlul *Desen* va fi desenat graficul ecuaţiei introduse pe o componentă de tip TImage.

13. Creaţi aplicaţia *Parabole*. Pe suprafaţa formei se plasează 4 butoane de comandă cu titlurile respectiv sus, jos, dreapta, stinga. La efectuarea unui click pe unul dintre butoane va fi desenată o parabolă cu ramurile orientate respectiv în sus, în jos, la dreapta, la stînga.

14. Elaboraţi o aplicaţie prin intermediul căreia va fi desenat sistemul cartezian de coordonate Oxy cu marcajele şi numerotările corespunzătoare. În acest sistem reprezentaţi punctul A şi proiecţiile acestuia pe axele de coordonate. Coordonatele punctului vor fi citite prin intermediul a două cutii de editare.

15. Creaţi aplicaţia Grafic. Prin intermediul acestei aplicaţii se vor desena graficele următoarelor funcţii:

$f(x) = x^3 + x^2$

$f(x) = \frac{1}{x} + \frac{1}{x^2}$

$f(x) = \frac{|x|}{x^2 + 1}$

$f(x) = x^2 \ln x$

$f(x) = \sin x + \frac{1}{2}\sin 2x$

$f(x) = x^2 + \frac{8}{x}$

$f(x) = e^{-x^2}$

$f(x) = \frac{e^x - e^{-x}}{2}$

$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

$f(x) = \frac{tgx}{x}$

16. Creaţi aplicaţia Roteşte triunghi. Pe suprafaţa formei se plasează două butoane cu titlurile respectiv Deseneaza triunghi şi Roteste triunghi. La efectuarea unui click pe primul buton se va desena un triunghi, iar la efectuarea unui click pe butonul al doilea va fi rotit triunghiul, citindu-se în prealabil măsura unghiului de rotire şi coordonatele punctului faţă de care se va efectua rotirea.

17. Creaţi aplicaţia Roteste patrulater. Pe suprafaţa formei se plasează două butoane cu titlurile respectiv Deseneaza patrulater şi Roteste patrulater. La efectuarea unui click pe primul buton se va desena un triunghi, iar la efectuarea unui click pe

butonul al doilea va fi rotit patrulaterul, citindu-se în prealabil măsura unghiului de rotire şi coordonatele punctului faţă de care se va efectua rotirea.

18. Creaţi aplicaţia Creion. Aplicaţia va simula componenta „Pencil” din aplicaţia PAINT. Prin intermediul unui meniu contextual va fi posibil de selectat culoarea creionului şi curăţirea suprafeţei de desenare.

19. Creaţi aplicaţia Deplasare. Pe suprafaţa formei este desenată o figură. La efectuarea unui click pe butonul cu titlul Start se va mişca figura pe traiectoria unui pătrat. Pentru aceasta se vor indica coordonatele pătratului (colţul stînga sus al pătratului şi lungimea laturii). Efectuarea unui click pe butonul Stop va stopa procesul de rotire a figurii. Prin intermediul butoanelor cu titlurile respectiv Mareste şi Micsoreaza se va mări, respectiv micşora viteza de deplasare a figurii.

20. Creaţi aplicaţia Plutire. Pe suprafaţa formei este desenată o figură (cerc, patrat, dreptunghi). La efectuarea unui click pe butonul cu titlul Start figura va ”pluti” pe suprafaţa formei (se va deplasa pe traiectoria unei sinusoide). Efectuarea unui click pe butonul Stop va stopa procesul de ”plutire” a figurii.

9. Componenta Timer

1. Elaboraţi o aplicaţie care va permite modificarea culoarea formei peste un interval de timp. Intervalul de timp se va introduce de către utilizator. Culoarea formei se va modifică în mod aleator dintr-o gamă de 8 culori.

2. Elaboraţi o aplicaţie care va modifica poziţia formei peste un interval de timp. Intervalul de timp se va introduce de către utilizator. Poziţionarea formei va fi efectuată în mod aleator astfel: sus, jos, dreapta, stînga, centru.

3. Creaţi aplicaţia *Logare*. Pe suprafaţa formei va fi plasat un buton cu titlul *Logare*, la efectuarea unui click pe suprafaţa acestuia se va afişa o nouă fereastră, unde utilizatorul va introduce numele şi parola. Pentru introducerea acestora se vor acorda 20 secunde. Dacă utilizatorul nu reuşeşte să introducă datele de logare corect, aplicaţia este închisă. Numele de utilizator şi parola corectă vor fi fixate în codul programului.

4. Creaţi aplicaţia *Tabla Inmulţirii*. Pe suprafaţa formei cu titlul *Tabla Inmultirii* este plasat butonul cu titlul *Verifica*. Efectuarea unui click pe suprafaţa acestuia va afişa o întrebare de genul $x*y= ?$, unde $1<x,y<11$. Pentru scrierea

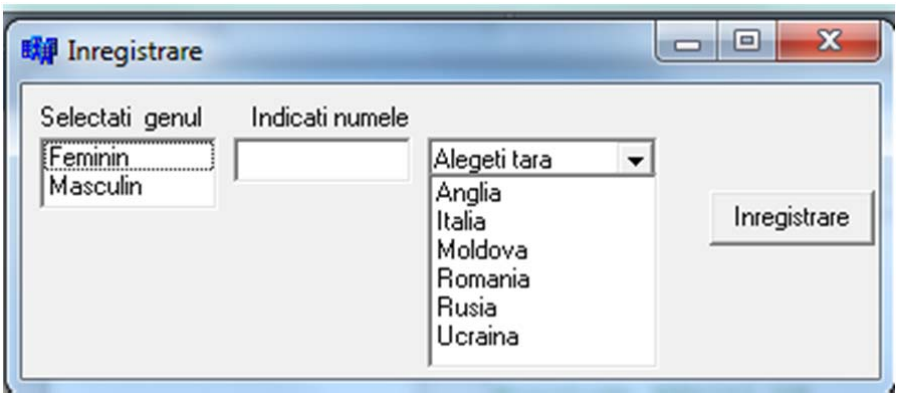
răspunsului corect utilizatorul va avea la dispoziţie 8 secunde. La următoarea întrebare, timpul de răspuns este micşorat cu 400 milisecunde. După expirarea timpului va fi afişat un mesaj prin care se va indica numărul de răspunsuri corecte.

5. Elaboraţi o aplicaţie care va simula un semafor. Prin intermediul cutiilor de editare se va introduce timpul pentru modificarea culorilor semaforului. Schimbarea uneia dintre culori va genera apariţia unui mesaj cu titlul corespunzător.

10. Aplicaţii cu liste de opţiuni şi casete combinate

1. Creaţi aplicaţia *Ţară/Capitală*. Într-o listă cu opţiuni se încarcă informaţii cu numele a 5 ţări. Într-o casetă combinată se încarcă informaţia despre capitalele acestor ţări. Utilizatorul va alege mai întîi ţara, după care capitala. La alegerea capitalei va fi afişată o fereastră în care se va indica, corectitudinea răspunsului.

2. Elaboraţi o aplicaţie care va permite unei persoane să se înregistreze. Forma va arăta în modul următor :



După selectarea genului, indicarea numelui şi alegerea ţării se va efectua un click pe butonul cu titlul *Inregistrare*. La efectuarea unui click pe suprafaţa acestuia se va afişa pe suprafaţa formei selecţia efectuată.

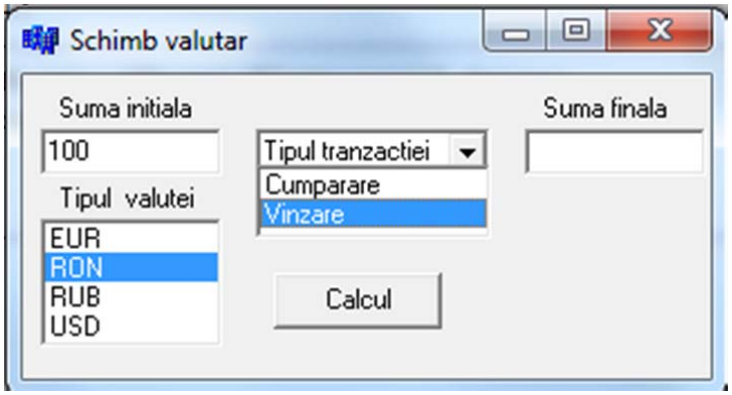
3. Să se elaboreze o aplicaţie care determină greutatea ideală a unei persoane cunoscînd înălţimea, vîrsta şi sexul persoanei. Vîrsta şi înălţimea va fi indicată prin intermediul componentelor *ComboBox*, iar sexul va fi indicat prin intermediul componentei *ListBox*. După indicarea datelor utilizatorul va efectua un click pe butonul *Determina*, care va afişa un mesaj cu greutatea ideală. Formulele de calcul sunt:

Gmasculin = 50 + 0.75 * (înălţime - 150) + (vîrsta - 20) / 4,

Gfeminin = Gmasculin – 10.

4. Elaboraţi o aplicaţie care va pemite determinarea numerelor prime pe segmentul [a,b]. Setarea valorilor *a* şi *b* se va efectua prin intermediul a două cutii text. După scrierea acestora se va efectua un click pe butonul *Setare*, valorile din cadrul acestui segment vor fi scrise într-o componetă *ComboBox*. La efectuarea unui click pe una dintre valori va fi afişat un mesaj în care se va indica dacă valoarea selectată este sau nu număr prim.

5. Creaţi aplicaţia *Schimb valutar*. Fişierul valuta.in conţine cursul valutar pentru ziua curentă în următorul format: *tip preţ_cumparare preţ_vînzare*. Aplicaţia va arăta astfel :



După indicarea sumei iniţiale şi a tipului acesteia se va indica tipul tranzacţiei. După efectuarea unui click pe butonul *Calcul* va fi scrisă suma finală, în dependenţă de cursul zilei (din fişierul *valuta.in*).

6. Creaţi aplicaţia *Magazin*. Fereastra aplicaţiei va arăta în modul următor:

Utilizatorul va alege produsul, apoi va scrie cantitatea în grame. Prin intermediul butonului cu titlul *Adauga in Cos* numele produsului şi cantitatea acestuia vor fi adăugate în componenta ListBox. La efectuarea unui click pe butonul cu titlul *Achita* se va afişa suma ce necesită a fi achitată pentru produsele selectate.

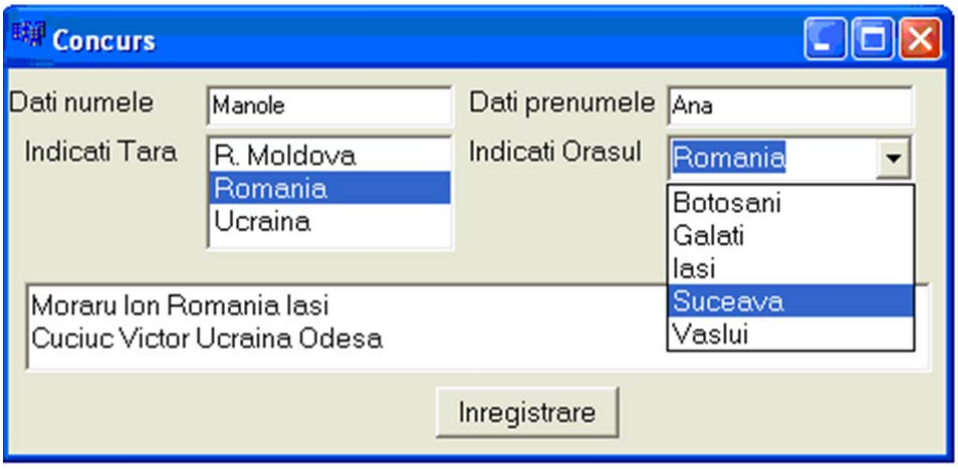
7. Elaboraţi o aplicaţie prin intermediul căreia se va modifica forma cursorului, culoarea formei şi stilul marginii ferestrei. Pentru elaborarea aplicaţiei pe suprafaţa formei vor fi plasate 3 componente ComboBox. În prima componentă ComboBox vor

fi introduse 5 valori pentru forma cursorului, în a doua componentă ComboBox vor fi scrise 5 valori pentru culoarea formei, iar în a treia componentă 5 valori ale proprietăţii BorderStyle. La efectuarea unui click pe una din opţiuni se va seta proprietatea respectivă. Pentru fiecare din componentele ComboBox va fi creat un meniu de tip TPopupMeniu cu opţiunile: adăugare şi excludere. Prin intermediul acestui meniu utilizatorul va avea posibilitatea de a adăuga valori, respectiv de a exclude.

8. Creaţi aplicaţia *Orarul Lecţiilor*. Într-o componentă ComboBox sunt scrise zilele săptămînii. La efectuarea unui click pe una din zilele săptămînii va fi afişat orarul lecţiilor pentru ziua respectivă. În cazul în care nu sunt ore în acea zi, va fi afişat mesajul „Zi libera”.

9. Creaţi aplicaţia *Divizori*. Prin intermediul a două cutii de editare se citesc numerele întregi *a* şi *b*. În componenta ComboBox se scriu toate numerele de pe segmentul [a;b]. La efectuarea unui click pe unul dintre numerele din listă vor fi afişaţi toţi divizorii numărului selectat.

10. Elaboraţi o aplicaţie care va permite înregistrarea participanţilor la concursul de Informatică. La concurs participă concurenţi din Republica Moldova, România, şi Ucraina. Din Republica Moldova vin participanţi din oraşele: Chişinău, Bălţi, Orhei, Soroca, Cahul, Edineţ. Din România vin participanţi din oraşele: Iaşi, Vaslui, Suceava, Botoşani, Galaţi. Din Ucraina vin participanţi din oraşele: Odesa, Doneţk, Herson, Hotin. Forma aplicaţiei va arăta în modul următor:



La efectuarea unui click pe butonul cu titlul Inregistrare datele despre concurent vor fi adăugate într-o componentă de tip TMemo. Pentru aceasta va fi creat un meniu de tip TPopupMenu, prin intermediul căruia va fi posibilă salvarea datelor într-un fişier, cît şi excluderea tuturor datelor despre concurenţi.

11. Componenta StringGrid

1. Creaţi aplicaţia *Vector*. Prin intermediul unei componente de tip TEdit şi TStringGrid se citesc elementele vectorului. Despre vector se vor afişa:

a) Numărul de elemente pare;
b) Numărul de elemente impare;
c) Elementul maxim;
d) Elementul minim;
e) Suma elementelor;
f) Media aritmetică.

2. Elaboraţi o aplicaţie prin intermediul căreia elementele unui vector vor fi sortate crescător şi descrescător.

3. Elaboraţi o aplicaţie care va prelucra elementele unei matrice cu *n* linii şi *m* coloane. Valorile *n* şi *m* vor fi citite prin intermediul a două componente de tip TEdit. Elementele matricei se citesc de la tastatură sau se generează de către calculator. Dimensiunea matricei se va citi prin intermediul cutiilor de editare. Pe suprafaţa formei vor fi prezentate următoarele informaţii despre matrice :

a) suma elementelor matricei;
b) elementul maxim;
c) elementul minim;
d) maximele pe linii;
e) maximele pe coloane;
f) minimele pe linii;
g) minimele pe coloane;
h) sumele pe linii;
i) sumele pe coloane;
j) perimtrul matricei.

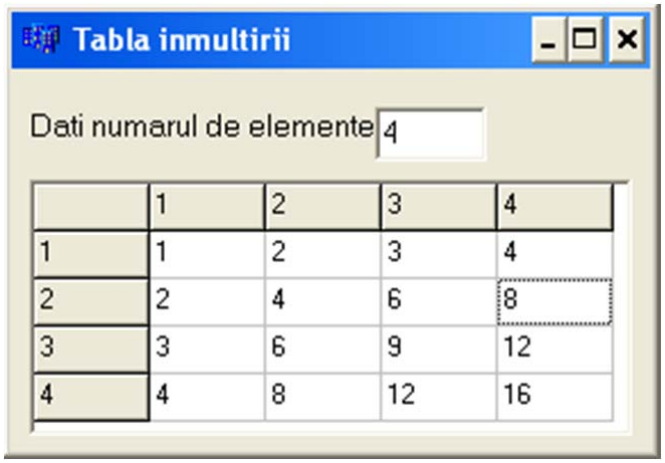
4. Elaboraţi o aplicaţie care va genera o matrice cu *n* linii şi *m* coloane. Elementele matricei vor fi scrise într-o componentă de tip TStringGrid. Valorile *n* şi *m* vor fi citite prin intermediul a două componente de tip TEdit. Pe suprafaţa formei vor fi plasate butoane radio prin intermediul cărora vor fi efectuate următoarele operaţii :

a) sortarea tuturor elementelor matricei (crescător sau descrescător);
b) sortarea elementelor matricei pe linii (crescător sau descrescător);
c) sortarea elementelor matricei pe coloane (crescător sau descrescător);

5. De la tastatură se introduc coeficienţii unui sistem de ecuaţii liniare de ordinul 3. Elaboraţi o aplicaţie care va permite determinarea soluţiilor acestui sistem.

6. De la tastatură se introduc dimensiunea şi elementele a două matrice pătratice B şi C. Elaboraţi o aplicaţie care va determina elementele matricei A, dacă A=B*C –2*(B+C).

7. Creaţi aplicaţia *Tabla înmulţirii*. De la tastatură se citeşte numărul elementelor din tabel. La efectuarea unui click pe butonul cu titlul *Calcul* va fi afişat tabelul corespunzător. De exemplu pentru 4 elemente tabelul va arăta în modul următor:

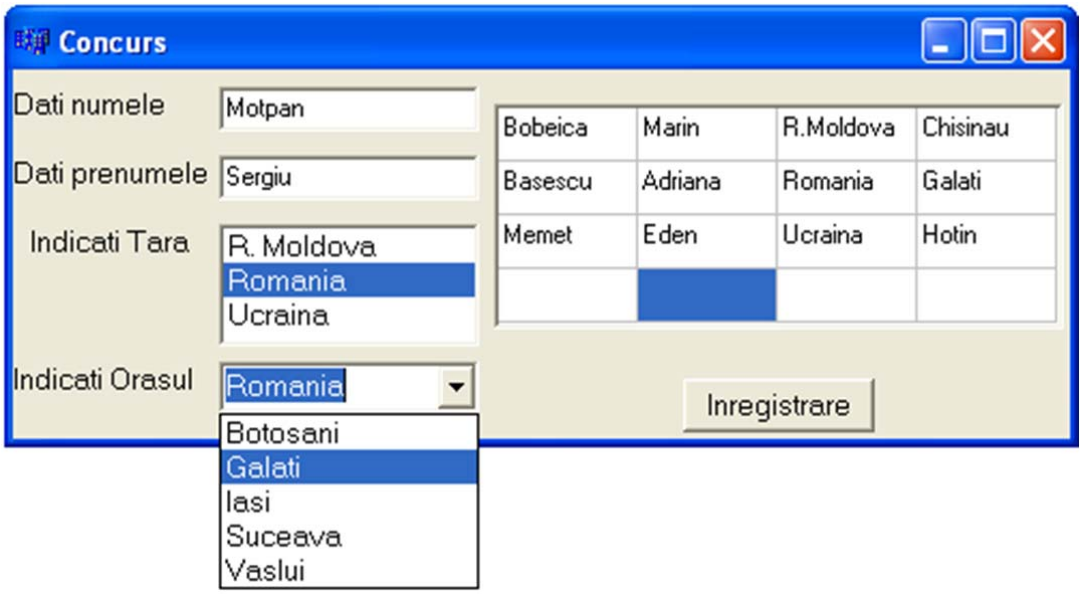


	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16

8. Creaţi aplicaţia *Evidenţa studenţilor*. Într-o componentă de tip TStringGrid se introduc datele despre studenţi şi anume: numele, prenumele, grupa, notele la 5 discipline. Aplicaţia va permite încărcarea acestor date din fişier. După indicarea acestora va fi calculată media pentru fiecare student. Elaboraţi o aplicaţie care va realiza un meniu cu următoarele opţiuni:

a) Studentul cu media maximă;
b) Studentul cu media minimă;
c) Media generală;
d) Procentul notelor de 9 şi 10;
e) Procentul notelor mai mici ca 5.

9. Elaboraţi o aplicaţie care va permite înregistrarea participanţilor la concursul de Informatică. La concurs participă concurenţi din Republica Moldova, România, şi Ucraina. Din Republica Moldova vin participanţi din oraşele: Chişinău, Bălţi, Orhei, Soroca, Cahul, Edineţ. Din România vin participanţi din oraşele: Iaşi, Vaslui, Suceava, Botoşani, Galaţi. Din Ucraina vin participanţi din oraşele: Odesa, Doneţk, Herson, Hotin. Forma aplicaţiei va arăta în modul următor:



La efectuarea unui click pe butonul cu titlul Inregistrare datele despre concurent vor fi adăugate într-o componentă de tip TStringGrid. De asemenea va fi creat un meniu cu opţiunile:

a) Total – va afişa numărul total de participanţi;
b) Tari – va afişa numărul de participanţi pentru fiecare ţară;
c) Oras – va afişa oraşul cu cei mai mulţi participanţi, în caz că sunt mai multe oraşe, se vor afişa denumirile acestora, urmat de numărul de participanţi.

10. Creaţi aplicaţia *Registru electronic*. Prin intermediul unei componente StringGrid se vor introduce datele despre studenţii unei grupe. Registrul electronic va oferi următoarele posibilităţi:

a) Determinarea mediei pentru fiecare student;
b) Determinarea mediei grupei;
c) Afişarea studenţilor neatestaţi (studenţii care au mai mult de 50% de absenţe), absenţa se va nota prin intermediul caracterului *a*;
d) Afişarea studenţilor restanţieri;
e) Afişarea studenţilor eminenţi.

12. Aplicaţii cu Data Calendaristică

1. Elaboraţi o aplicaţie care va permite afişarea informaţiei despre data şi timpul curent. Informaţia va fi prezentată conform exemplului:

Vineri 30 octombrie anul 2009 ora 10 si 31 de minute si 14 secunde.

2. Creaţi aplicaţia *Alarmă*. Pe suprafaţa formei va fi creat un ceas digital. Prin intermediul butonul *Seteaza*, se va introduce timpul pentru alarmă. În cazul în care timpul curent va coincide cu timpul indicat în cazul alarmei, culoarea formei va

deveni roşie, iar pe suprafaţa ei va apărea textul *Alarma* de culoare neagră cu mărimea 72pt.

3. Elaboraţi o aplicaţie care va permite închiderea aplicaţiei peste un interval de timp *t*. Intervalul de timp se va introduce de la tastatură. Pentru introducerea timpului vor fi utilizate 3 butoane radio care vor avea posibilitatea de a introduce timpul în minute, în secunde şi în ore.

4. Creaţi aplicaţia *Tastare*. Pe suprafaţa formei vor fi plasate două butoane cu titlurile *Lanseaza test* şi *Verifică test*. La efectuarea unui click pe primul buton se for afişa *n* caractere (vor fi afişate în mod aleator caractere ale alfabetului latin), unde *n* se citeşte de la tastatură. Prin intermediul unei cutii de editare utilizatorul va tasta caracterele afişate După introducerea corectă a tuturor caracterelor se va afişa timpul necesar pentru tastarea lor.

5. Creaţi aplicaţia *Test*. Pe suprafaţa formei vor fi plasate două butoane de comandă cu titlurile respectiv: *Lanseaza test* şi *Verifică test*. La efectuarea unui click pe primul buton se vor afişa 5 întrebări, fiecare din ele oferind 5 variante de răspuns. După indicarea răspunsurilor se va efectua un click pe butonul cu titlul *Verifică test*. Efectuarea unui click pe suprafaţa acestuia va afişa numărul de răspunsuri corecte şi timpul necesar pentru indicarea răspunsurilor.

6. Creaţi aplicaţia *Calculator*. Prin intermediul acestei aplicaţii vor fi efectuate operaţii cu data calendaristică: adunare, scădere, comparaţie, transformare (din dată în zile, saptămîni, luni şi reciproc).

7. Creaţi aplicaţia *Orarul studentului*. În dosarul proiectului vor fi create şapte fişiere cu titlurile respectiv: *luni.txt, ..., duminica.txt,* în fiecare din ele fiind scris orarul lecţiilor pentru ziua respectivă. Prin intermediul unei componente de tip TMonthCalendar utilizatorul va alege o dată. După alegerea datei într-o componentă de tip TMemo se va afişa orarul pentru data aleasă. Se va ţine cont de faptul că în perioadele 31 decembrie - 15 ianuarie şi 24 iunie - 31 august studentul se află în vacanţă, iar în perioada 15 - decembrie 30 decembrie şi 1 iunie - 13 iunie studentul se află în sesiune.

8. Creaţi aplicaţia *Agenda*. Utilizatorul va indica data şi ora cînd urmează a avea loc un eveniment. Se va permite adăugarea a mai multor evenimente. Pe suprafaţa formei se va afişa un mesaj corespunzător dacă au rămas mai puţin de 24 ore pînă la evenimentul respectiv.

9. Elaboraţi o aplicaţie care va permite utilizatorului să determine zodia în care s-a născut şi numărul de zile rămase pînă la data naşterii. Setarea zilei de naştere se va efectua prin intermediul unei componente corespunzătoare (TMonthCalendar). După selectarea acesteia utilizatorului i se va afişa informaţia respectivă.

10. Creaţi aplicaţia *Ceas digital*. La mijlocul ferestrei va fi afişat un ceas digital ce indică timpul în formatul *ore:min:sec*, cu mărimea de 72 pt. Culoarea formei va fi modificată dintr-o gamă de 5 culori la fiecare minut. Prin intermediul unui meniu utilizatorul va putea seta ora, minutele, respectiv secundele.

13. Aplicaţii cu directoare şi fişiere

1. Elaboraţi o aplicaţie care va permite crearea a *n* directoare într-un directoriu indicat. Directoriul va fi indicat prin intermediul unei componente specifice, în caz particular *DirectoryListBox*. Valoarea *n* va fi citită de la tastatură.

2. Elaboraţi o aplicaţie care va permite crearea a *n* fişiere textuale într-un directoriu indicat. Directoriul va fi indicat prin intermediul unei componente specifice, în caz particular *DirectoryListBox*. Valoarea *n* va fi citită de la tastatură.

Indicaţie: se va utiliza funcţia *FileCreate(nume fişier)*.

3. Elaboraţi o aplicaţie care va permite afişarea fişierelor dintr-un directoriu indicat, care au fost create în săptămîna curentă. Directoriul va fi indicat prin intermediul unei componente specifice, în caz particular *DirectoryListBox*.

4. Elaboraţi o aplicaţie care va permite afişarea fişierelor, dintr-un directoriu indicat, memoria cărora nu depăşeşte *n KB*, unde *n* se citeşte de la tastatură. Directoriul va fi indicat prin intermediul unei componente specifice, în caz particular *DirectoryListBox*.

5. Elaboraţi o aplicaţie care va informa utilizatorul despre fişierul ales. Prin intermediul unei componente specifice utilizatorul selectează un fişier. După selectarea acestuia utilizatorului îi va fi prezentată informaţia despre fişierul dat şi anume :

a) timpul creării *FileDateToDateTime(nume_fisier.Time)* ;
b) mărimea fişierului;
c) tipul fişierului (ascuns, de sistem, arhivat, obişnuit);
d) adresa fişierului.

6. Elaboraţi o aplicaţie prin intermediul căreia i se va oferi utilizatorului să „navigheze” prin directoarele şi fişierele stocate pe calculator. La efectuarea unui double-click pe un fişier cu una dintre extensiile (*.txt, .cpp, .h, .pas*), conţinutul acelui fişier va fi afişat prin intermediul unei componente de tip TMemo.

7. Elaboraţi o aplicaţie care va permite utilizatorului să selecteze unul sau mai multe fişiere pentru redenumirea acestora.

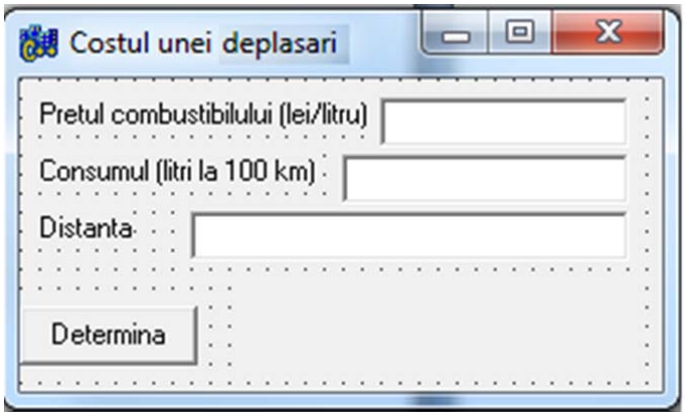
8. Creaţi aplicaţia *Detalii despre fişiere*. Prin intermediul unei componente *DirectoryListBox* utilizatorul va selecta un dosar. După alegerea dosarului prin intermediul unei componente de tip TStringGrid se va afişa numele fişierelor şi mărimile acestora. De asemenea, se va afişa numărul total de fişiere şi mărimea totală a acestora.

9. Elaboraţi o aplicaţie care va permite afişarea fişierelor dintr-un directoriu selectat (se vor utiliza componentele *DirectoryListBox, FileListBox*). Prin intermediul a două cutii de editare se va permite înlocuirea caracterelor din numele fişierului. Nu se va permite modificarea extensiei fişierului.

10. Elaboraţi o aplicaţie care va permite vizualizarea fişierelor dintr-un directoriu în perioada de timp *a-b,* unde *a,b* sunt de tipul TDateTime. Valorile *a* şi *b* se vor citi prin intermediul componentelor de tip TDateTimePicker. Directoriul va fi indicat prin intermediul componentei de tip TDirectoryListBox. Numele fişierelor vor fi scrise într-o componentă de tip TStringGrid. Se va afişa şi data creării fişierului.

14. Aplicaţii diverse

1. Elaboraţi o aplicaţie, prin intermediul căreia se va determina costul unei deplasări cu automobilul. Aplicaţia va arăta astfel :



2. Elaboraţi o aplicaţie prin intermediul căreia va fi posibil de calculat preţul unei convorbiri telefonice, dacă preţul unei convorbiri în reţea costă 75 bani/min, cu alt operator naţional costă 1,76 lei/min, iar o convorbire peste hotare costă 2,5 lei/min. Aplicaţia va arăta astfel :

3. Elaboraţi o aplicaţie care va permite calculul costului total de cazare a unei persoane la un hotel. Persoana se va înregistra la hotel în baza actului de identitate. De asemenea se va păstra data înregistrării. Persoana va achita următoarele servicii:

Cazare – fiecare tip de cameră va avea un anumit cost pentru o zi;

Hrana – persoana va avea de ales unul din trei meniuri pentru o zi: econom, clasic sau lux.

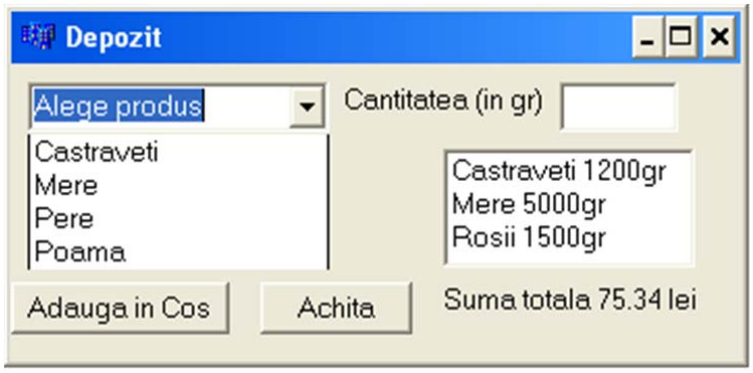
La sfîrşitul perioadei de cazare persoanei i se va calcula suma care trebuie să fie achitată. Persoana va beneficia de reduceri în următoarele cazuri :

a) Perioada de cazare de 4-7 zile oferă o reducere de 3% din suma totală;
b) Perioada de cazare de 8-14 zile oferă o reducere de 7% din suma totală;
c) Perioada de cazare de 15-30 zile oferă o reducere de 10% din suma totală;
d) Perioada de cazare mai mare de 30 zile oferă o reducere de 15% din suma totală.

4. Elaboraţi o aplicaţie prin intermediul căreia utilizatorul va avea posibilitatea de a-şi determina plata serviciilor comunale pentru luna curentă. Cu ajutorul aplicaţiei se vor determina costul la gaz, apă caldă, apă rece şi electricitate. Se recomandă a utiliza componenta TabbedNotebook.

5. Creaţi aplicaţia *Milionar*. Această aplicaţie va simula jocul *Cine vrea să devină MILIONAR ?* Pentru fiecare întrebare utilizatorul va avea la dispoziţie un minut pentru a da răspunsul corect.

6. Creaţi aplicaţia *Depozit*. Fereastra aplicaţiei va arăta în modul următor:



Utilizatorul va alege produsul, apoi va scrie cantitatea în grame. Prin intermediul butonului cu titlul *Adauga in Cos* numele produsului şi cantitatea acestuia vor fi adăugate în componenta ListBox. La efectuarea unui click pe butonul cu titlul *Achita* se va afişa suma ce necesită a fi achitată pentru produsele selectate. Pentru Componenta ComboBox va fi creat un meniu contextual cu 2 opţiuni:

a) adaugă – pentru adăugarea unui produs în depozit într-o anumită cantitate;
b) exclude – pentru excluderea unui produs din depozit.

Pentru componenta ListBox va fi creat un meniu contextual cu opţiunea exclude, pentru a oferi utilizatorului posibilitatea de a exclude un produs.

7. Elaboraţi un proiect care va permite vizualizarea imaginilor dintr-un directoriu indicat. Iniţial va permite vizualizarea din directoriul curent, ca ulterior să permită modificarea acestuia. Aplicaţia dată va conţine instrumente pentru mărirea/micşorarea imaginilor, cît şi pentru a trece la următoare/precedenta imagine.

8. Elaboraţi un proiect care va simula un editor de texte similar aplicaţiei Notepad.

9. Elaboraţi un proiect care va simula un editor de imagini (similar aplicaţiei Paint). Editorul va conţine instrumente specifice pentru desenarea figurilor. De asemenea va permite încărcarea şi salvarea unei imagini.

10. Creaţi aplicaţia *Calculator*. Prin intermediul aplicaţiei se va oferi utilizatorului posibilitatea de a efectua:

a) operaţii cu numere reale;
b) operaţii de conversie dintr-un sistem de numeraţie în altul;
c) operaţii trigonometrice;
d) operaţii logaritmice.

11. Elaboraţi un proiect prin intermediul căruia se vor efectua operaţii asupra tipului de date *arbori binari de căutare*. În calitate de noduri ale arborelui binar de căutare vor fi numere întregi. Prin intermediul unui meniu se vor efectua operaţiile:

a) crearea arborelui binar de căutare (încărcarea datelor din fişier);
b) desenarea arborelui pe suprafaţa formei;
c) parcurgerea arborelui prin metodele: inordine, preordine şi postordine;
d) căutarea unui nod;
e) inserarea unui nod în arbore;
f) excluderea unui nod din arbore.

12. Elaboraţi un proiect pentru gestiunea medicamentelor unei farmacii. Informaţiile pentru medicamentele din farmacii sunt: numele medicamentului, preţ, cantitate, data primirii, data expirării. Evidenţa medicamentelor se va efectua printr-o structură de date de tip arbore de căutare. Aplicaţia va conţine un meniu prin intermediul căruia se vor efectua următoarele operaţii:

a) încărcarea datelor despre medicamente din fişier şi prin intermediul unui formular destinat introducerii datelor;
b) căutarea unui medicament după nume;
c) afişarea medicamentelor în ordine lexicografică a numelui (va fi utilizată o componentă Memo);
d) eliminarea unui medicament prin indicarea numelui acestuia;
e) afişarea medicamentelor care au data de expirare mai mică decît o dată specificată.

13. Elaboraţi o aplicaţie prin intermediul căreia va fi rezolvat jocul SUDOKU. Prin intermediul componentei StringGrid utilizatorul va introduce numerele în cutie. Efectuarea unui click pe suprafaţa butonului cu titlul *Rezolvare*, va rezolva jocul.

14. Elaboraţi aplicaţia *Operaţii cu mulţimi*. Cu ajutorul a două componente StringGrid se citesc elementele a două mulţimi A şi B. Prin intermediul unui meniu utilizatorului i se va oferi posibilitatea de a efectua operaţii cu mulţimi. Meniul va conţine opţiunile:

a) reuniune – va afişa elementele mulţimii $A \cup B$;
b) intersecţie – va afişa elementele mulţimii $A \cap B$;
c) diferenţa – va afişa elementele mulţimii $A \setminus B$;
d) incluziune element – va verifica dacă un element (va fi introdus prin intermediul unei cutii de editare) se include în mulţimea A;
e) incluziune mulţimi – va verifica dacă mulţimea A se include în mulţimea B;

15. Creaţi aplicaţia *Calculator cu numere mari*. Prin intermediul aplicaţiei se va oferi utilizatorului posibilitatea de a efectua operaţii cu numere întregi foarte mari (un număr va conţine pînă la 255 de cifre). Se vor efectua următoarele operaţii:

a) operaţii aritmetice (suma, diferenţa, produsul, cîtul şi restul împărţirii) între două numere foarte mari;
b) operaţii de comparaţie.

16. Creaţi aplicaţia *Matrice cu numere complexe*. Prin intermediul aplicaţiei se va oferi utilizatorului posibilitatea de a efectua operaţii cu o matrice pătratică elementele cărora sunt numere complexe. Elementele matricei se vor scrie într-o componentă StringGrid. Utilizatorului i se va oferi posibilitatea de a determina:

a) suma elementelor matricei;
b) maximele pe linii;
c) maximele pe coloane;
d) minimele pe linii;
e) minimele pe coloane;
f) sumele pe linii;
g) sumele pe coloane;
h) perimetrul matricei.

17. Elaboraţi aplicaţia *figuri geometrice*. Prin intermediul unui formular utilizatorul va avea posibilitatea a citi date despre următoarele figuri geometrice: cerc, triunghi echilateral, triunghi înscris în cerc. Despre fiecare figură se vor citi datele despre coordonatele centrului figurii, lungimea razei (laturii) şi culoarea. Se va crea o listă care va conţine datele despre mai multe figuri. Prin intermediul unui meniu utilizatorului i se va oferi posibilităţile:

a) adăugare de figuri noi (cerc, triunghi echilateral, triunghi înscris în cerc);
b) afişare a figurilor;
c) excludere a figurilor.

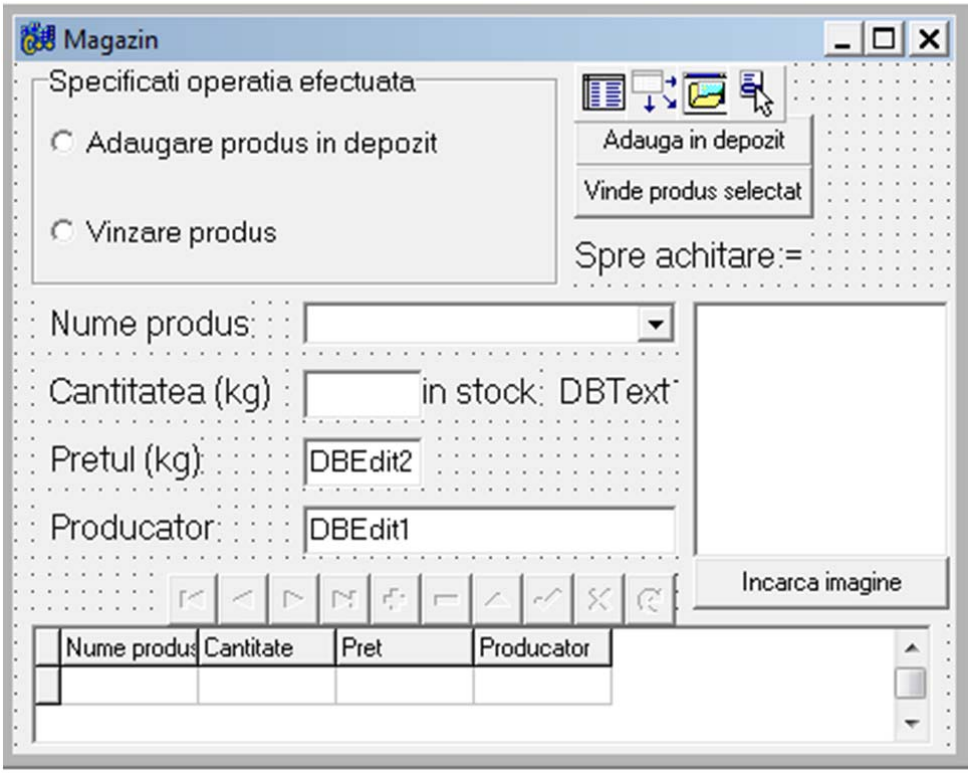
Capitolul III Baze de date

Mediul Borland C++ Builder oferă posibilitatea creării bazelor de date. Crearea lor se efectuează cu ajutorul utilitarelor *BDE Administrator* (pentru crearea aliasului) şi *Database Desktop* (pentru crearea tabelelor). Pentru crearea tabelelor va fi utilizat aliasul *BCDEMOS* în care vor fi create tabele şi interogări.

Probleme rezolvate

Problema1 Magazin

Elaboraţi o aplicaţie prin intermediul căreia va fi simulată activitatea unui magazin. Astfel, se vor efectua operaţii de primire a mărfii şi de vînzare a acesteia. Forma aplicaţiei va arăta în modul următor:



Realizare:

Pentru păstrarea datelor despre produsele din magazin vom crea tabelul *produse.db*, pe care îl vom salva în dosarul curent. Tabelul va avea următoarea structură:

Field Name	Type	Size	Key
Nume produs	A	25	
Producator	A	20	
Cantitate	N		
Pret	N		
Foto	G		

Pe suprafaţa formei plasăm componentele:

Button1 – pentru încărcarea imaginii, în cadrul proprietăţii *Caption* scriem textul *Incarca imagine*;

Button2 – pentru adăugarea unui produs în depozit, în cadrul proprietăţii *Caption* scriem textul *Adauga in depozit*;

Button3 – pentru vînzarea unui produs, în cadrul proprietăţii *Caption* scriem textul *Vinde produs selectat*;

ComboBox1 – pentru selectarea produsului;

Edit1 – pentru indicarea cantităţii;

RadioGroup1 – pentru indicarea operaţiei ce necesită a fi efectuată, setăm proprietăţile conform figurii;

OpenPictureDialog – pentru încărcarea imaginii în tabel, setăm proprietatea *Stretch* la *true*;

PopupMenu1 – creăm meniul *Adauga produs nou*, pentru a oferi posibilitatea de a adăuga produse noi;

Table1– setăm proprietatea *Active* la *true*;

DataSource1: DataSet=Table1;

DBGrid1: DataSource= DataSource1;

DBNavigator1: DataSource= DataSource1;

DBEdit1: DataSource= DataSource1, DataField=Producator;

DBEdit2: DataSource= DataSource1, DataField=Pret;

DBText1: DataSource= DataSource1, DataField=Cantitate;

DBImage1: DataSource= DataSource1, DataField=Foto;

Declaraţii:

```
int index, tip;
bool ad=true;double suma=0.0;
```

Prelucrăm evenimentul: OnCreate

```
Button1->Hide();
```

Prelucrăm evenimentul: *RadioGroup1Click*

```
tip=RadioGroup1->ItemIndex;
if(tip==1){Button1->Hide();ComboBox1->PopupMenu=NULL;}
else {Button1->Show();ComboBox1->PopupMenu=PopupMenu1;}
```

Prelucrăm evenimentul: *FormShow*

```
AnsiString s;Table1->First();
ComboBox1->Clear();
```

```
while(!Table1->Eof){
s= AnsiString(Table1->FieldValues["Nume produs"]);
ComboBox1->Items->Add(s);
Table1->Next();}
```

Prelucrăm evenimentul: *ComboBox1Change*

```
index=ComboBox1->ItemIndex;
Table1->MoveBy(index-1);
```

Prelucrăm evenimentul: *Button1Click*

```
OpenPictureDialog1->Execute();
AnsiString nume=OpenPictureDialog1->FileName;
if(nume.Length()>3){
 DBImage1->Picture->LoadFromFile(nume);
 DBImage1->CopyToClipboard();
 DBImage1->PasteFromClipboard();
}
```

Prelucrăm evenimentul: *Button2Click*

```
double d;
if(ad){
 Table1->MoveBy(index-1);
 Table1->Edit();
 d=Table1->FieldValues["Cantitate"];
 Table1->FieldValues["Cantitate"]= d+Edit1->Text.ToDouble();
}
else{Table1->Edit();
 Table1->FieldValues["Cantitate"]=Edit1->Text.ToDouble();
}
Table1->Post();
Edit1->Clear();
```

Prelucrăm evenimentul: *Adaugaprodusnou1Click*

```
Table1->Insert();
AnsiString s=InputBox("Adaugare produs","Specificati numele
produsului","");
ComboBox1->Items->Add(s);
Table1->FieldValues["Nume produs"]=s;
ad=false;
```

Prelucrăm evenimentul: *Button3Click*

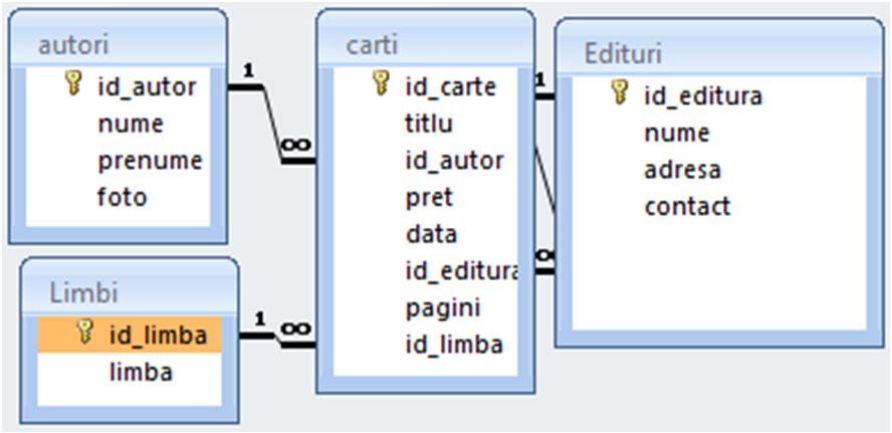
```
double c;
Table1->MoveBy(index-1);
Table1->Edit();
if(Table1->FieldValues["Cantitate"]<Edit1->Text.ToDouble())
Edit1->Text=Table1->FieldValues["Cantitate"];
c=Table1->FieldValues["Cantitate"]-Edit1->Text.ToDouble();
Table1->FieldValues["Cantitate"]=c;
suma=suma+Table1->FieldValues["pret"]*Edit1->Text.ToDouble();
Label6->Caption="Spre achitare:="+AnsiString(suma);
Edit1->Clear(); Table1->First();
while(!Table1->Eof){
 if(Table1->FieldValues["Cantitate"]==0) Table1->Delete();
 Table1->Next();
}
```

Prelucrăm evenimentul: *Table1AfterDelete*

```
AnsiString s;
Table1->First();
ComboBox1->Clear();
while(!Table1->Eof){
 s=AnsiString(Table1->FieldValues["Nume produs"]);
 ComboBox1->Items->Add(s);
 Table1->Next();
}
```

Problema2 Evidenţa cărţilor

Elaboraţi un proiect BCB prin intermediul căruia se va duce evidenţa cărţilor unei biblioteci, dacă schema bazei de date are următoarea structură:



Realizare:

Creem prin intermediul utilitarului Database Desktop tabelele de mai sus. În cadrul acestei aplicaţii vom utiliza mai multe forme. Pe forma principală plasăm componentele:

ADOConnection1 – facem conexiunea cu baza de date şi setăm proprietatea *Connected* cu valoarea *true;*

ADOTable1 *– setăm următoarele proprietăţi:*

Connection= ADOConnection1;

TableName=Carti;

Active=true;

Name=Carti;

Procedăm similar şi în cazul componentelor *ADOTable2, ADOTable3, ADOTable4* modificînd proprietăţile *Name* şi *TableName* corespunzător în *Autori*, *Edituri*, *Limbi*;

DataSource1 *– setăm următoarele proprietăţi:*

DataSet=Carti, modificăm valoarea proprietăţii *Name* cu *Carte*, procedăm similar şi în cazul următoarelor componente *DataSource2, DataSource3, DataSource4* cu valorile proprietăţii *Name* corespunzător *Autor*, *Editura*, *Limba*;

Image1 *– setăm următoarele proprietăţi:*

Picture – încărcăm o imagine;

Stretch=true;

Align=alClient;

ADOQuery1 *– setăm următoarele proprietăţi:*

Connection= ADOConnection1;

Name=Query;

Pentru forma principală setăm următoarele proprietăţi:

Name=Main;

Align=alClient;

mai adăugăm în proiect încă 7 forme, modificînd proprietăţile *Name* corespunzător: *Ad_autor*, *Ad_carte*, *Ad_limba*, *Ad_editura*, *Admin*, *Raport*;

Setăm *Position=poScreenCenter* pentru toate formele;

Includem în antetul fiecărui fişier **.cpp* fişierele header (*unit1.h – unit8.h*), după necesitate;

În fişierul *Unit1.h* la secţiunea **public:** declarăm variabila

AnsiString sql;

Prelucrăm evenimentul *OnCreate* al formei:

sql="SELECT Carti.titlu, Carti.pret, Carti.pagini,Carti.data, Editura.nume, Autori.nume, Autori.prenume, Limba.limba FROM Autori,Carti,Editura,Limba WHERE (Carti.id_autor = Autori.id_autor) AND (Editura.id_editura = Carti.id_editura) AND (Limba.id_limba = Carti.id_limba)";

MainMenu1 **–** meniul va conţine următoarele opţiuni:

Adauga	**Cauta**	**Tipar**	**Administrare**
Carte	Dupa titlu	Toate cartile	
Autor	Dupa limba	Liste cu autori	
Editura	Dupa pret		
Limba	Dupa data		

Se va plasa componenta ***PopupMenu1*** cu opţiunea *Adauga autor*,

DBLookupComboBox1:

DataSource=Main->Carte; DataField=id_autor; ListSource=Main->Autor; ListField=nume; KeyField=id_autor;PopupMenu=PopupMenu1;

Se va plasa componenta ***PopupMenu2*** cu opţiunea *Adauga limba;*

DBLookupComboBox2:

DataSource=Main->Carte; DataField=id_limba; ListSource=Main->limba; ListField=limba; KeyField=id_limba;PopupMenu=PopupMenu2;

Se va plasa componenta ***PopupMenu3*** cu opţiunea *Adauga editura;*

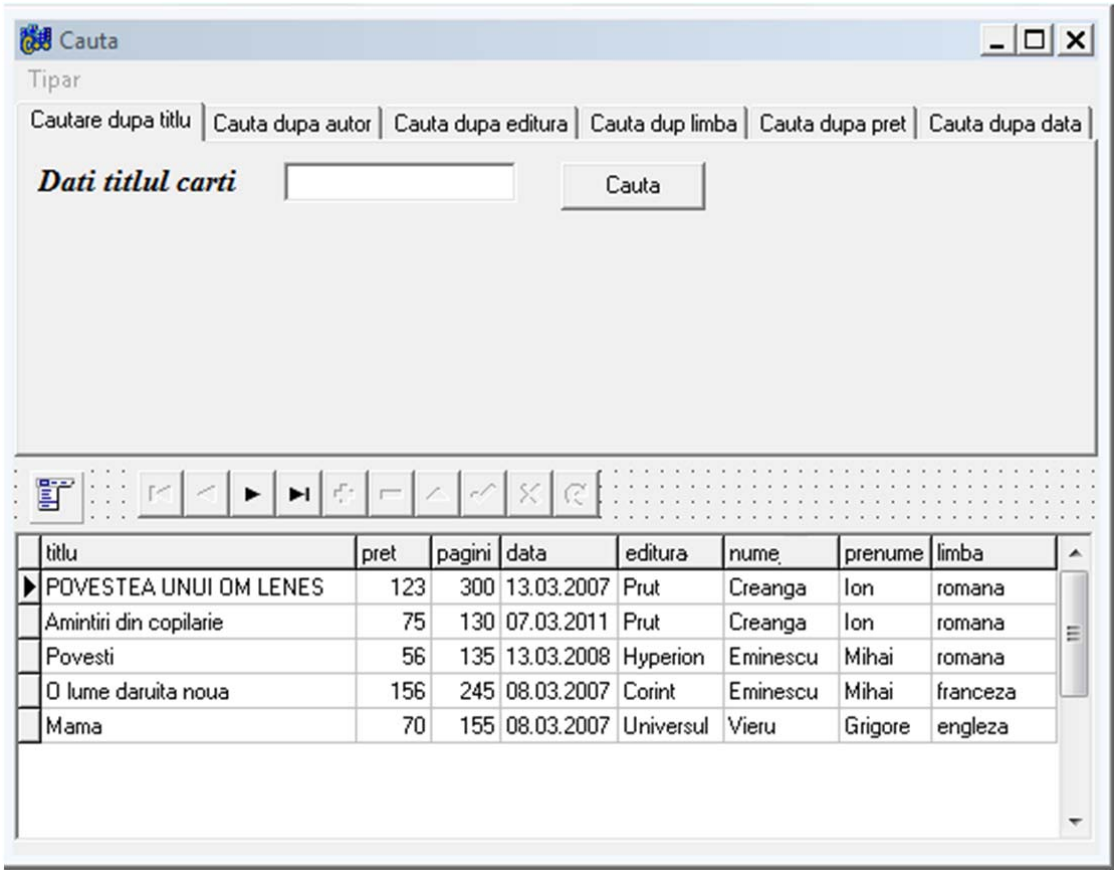
DBLookupComboBox3: *DataSource=Main->Carte; DataField=id_editura; ListSource=Main->editura;ListField=nume; KeyField=id_editura;PopupMenu=PopupMenu3;*

DateTimePicker1: - pentru selectarea datei.

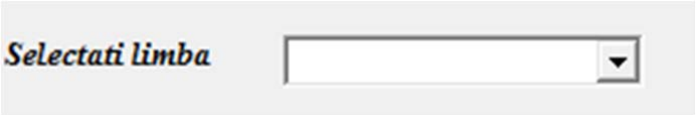
FORMULARUL cauta

Pe suprafaţa acestei forme vom crea un meniu principal numit *Tipar* (componenta MainMenu1), DBNavigator1, DBGrid1. Plasăm componenta TabbedNotebook1 şi creăm 4 pagini.

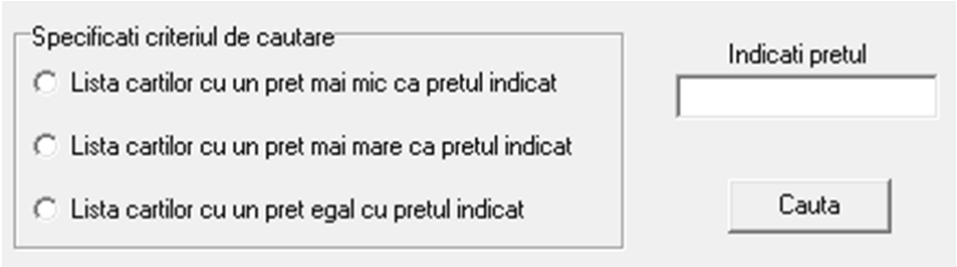
Prima pagina – căutare după titlu, vom utiliza componentele: Label1, Edit1, Button1;



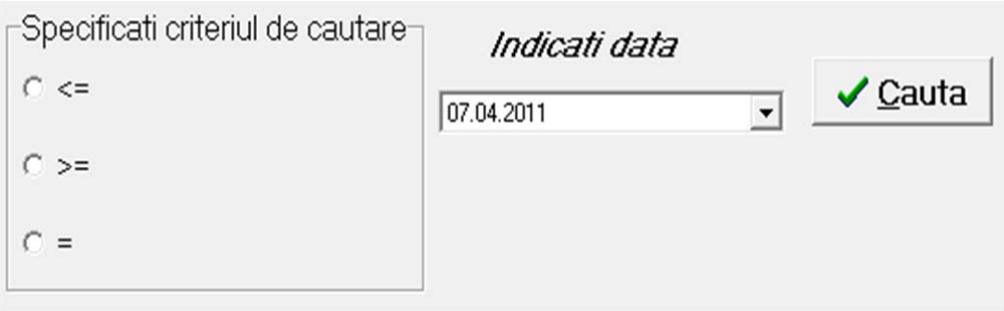
Pagina doi – căutare după limbă, vom utiliza componentele: *ComboBox1*;



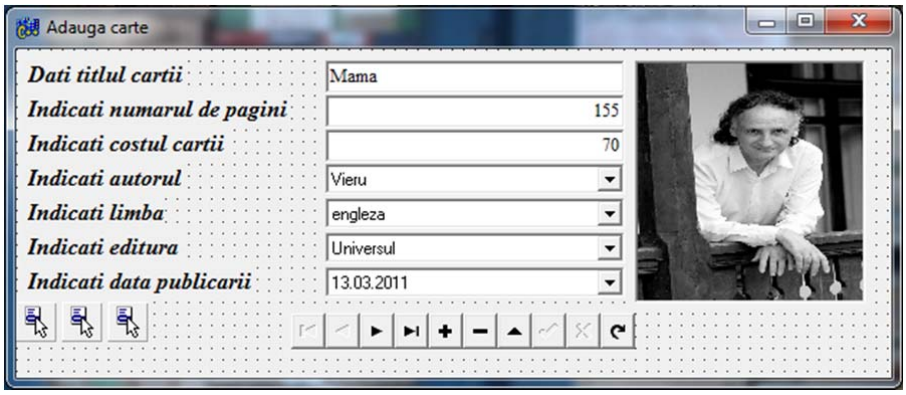
Pagina trei – căutare după preţ, vom utiliza componentele: *RadioGroup1*, *Edit2*, *Button2*;



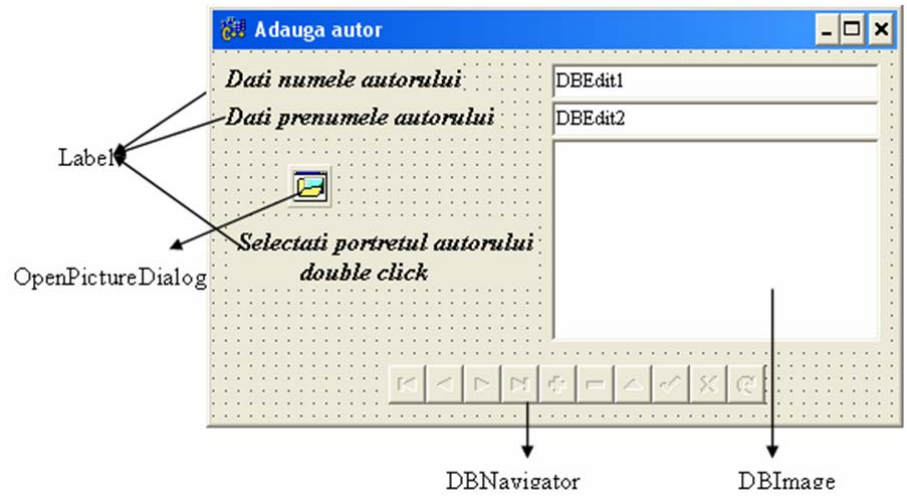
Pagina patru – căutare după data publicării, vom utiliza componentele: *RadioGroup2*, *DateTimePicker1*, *BitBtn1*;



FORMULARUL Ad_carte



FORMULARUL Ad_autor



Pe suprafaţa formei AD_autor plasăm componentele:

DBEdit1: *DataSource=Main->Autor; DataField=nume;*

DBEdit2: *DataSource=Main->Autor; DataField=prenume;*

DBImage1: *DataSource=Main->Autor; DataField=foto;*

DBNavigator1:*DataSource=Main->Autor;*

FORMULARUL Ad_limba

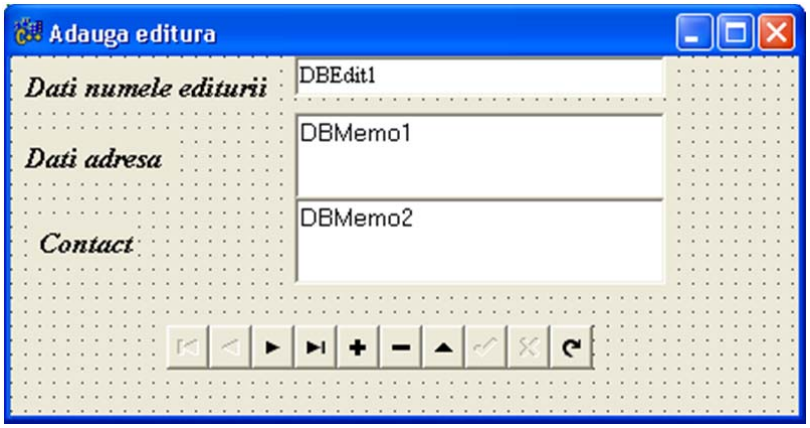
Pe suprafaţa formei AD_limba plasăm componentele:

Label1: *Caption=Specificati limba;*

DBEdit1: *DataSource=Main->Limba; DataField=limba;*

DBNavigator1: *DataSource=Main->Limba;*

FORMULARUL Ad_editura



Pe suprafaţa formei Ad_editura plasăm componentele:

DBEdit1: *DataSource=Main->Editura;DataField=nume;*

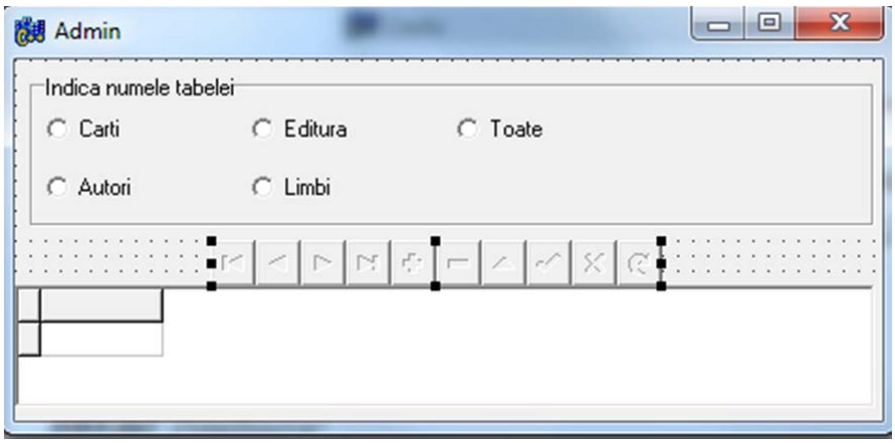
DBMemo1: *DataSource=Main->Editura; DataField=adresa;*

DBMemo2: *DataSource=Main->Editura; DataField=contact;*

DBNavigator1:*DataSource=Main->Editura;*

Label1, ***Label2, Label3*** corespunzător imaginii

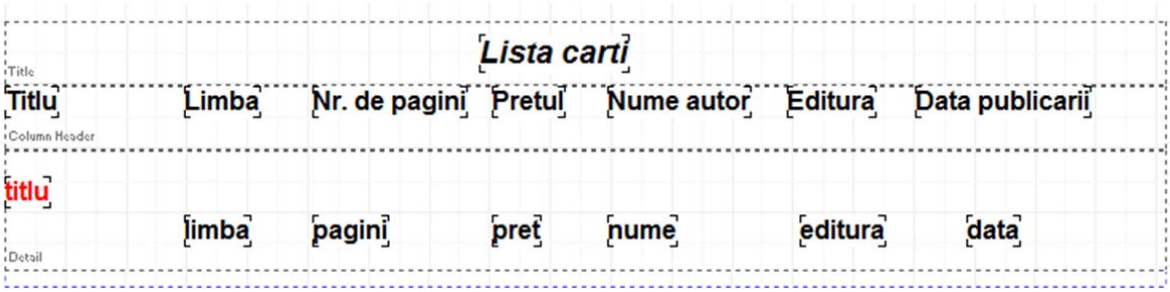
FORMULARUL Administrare



Pe suprafaţa formei *Admin* plasăm componentele:
RadioGroup1: *Columns=3; Caption=Indica numele tabelei;*
Items adăugăm cîmpurile corespunzător figurii;
DBNavigator1, ***DBGrid1***

FORMULARUL Raport

Pe forma Raport plasăm componenta *QuickRep1*, modificăm proprietatea *Name* în *Lista_carti.* Scriem textul interogării în cadrul componentei *Query* de pe forma principală (în cadrul proprietăţii *SQL* scriem interogarea, după care setăm *Active=true*;



Quickrep1: *Dataset=Main->Query;Bands->HasTitle=true;*
Bands->HasColumnHeader=true;
Bands->HasDetail=true;
Pe banda Title plasăm componenta *QRLabel1* şi atribuim proprietăţii *Caption* textul *Lista carti;*
Procedăm similar şi în cazul benzii *Column Header;*Pe banda *Detail* plasăm componentele:
QRDBText1*: Dataset=Main->Query;DataField=titlu;*
QRDBText2*: Dataset=Main->Query; DataField=limba;*
QRDBText3*: Dataset=Main->Query; DataField=pagini;*
QRDBText4*: Dataset=Main->Query; DataField=pret;*
QRDBText5*: Dataset=Main->Query; DataField=nume;*
QRDBText6*: Dataset=Main->Query; DataField=editura;*
QRDBText7*: Dataset=Main->Query; DataField=data.*

În cadrul acestei forme urmează a fi create mai multe rapoarte, plasăm componenta *Quickrep2* pe formă, iar pe suprafaţa acesteia:

Quickrep2: *Dataset=Main->Autori, Name=Lista_autori,*
Bands->HasTitle=true,
Bands->HasColumnHeader=true, Bands->HasDetail=true;
Pe banda Title plasăm componenta *QRLabel1* şi atribuim proprietăţii *Caption* textul *Lista autori;*
Procedăm similar şi în cazul benzii *Column Header;*
Pe banda *Detail* plasăm componentele:
QRDBText1*: Dataset=Main->Autori;DataField=nume;*
QRDBText2*: Dataset=Main->Autori; DataField=prenume;*
QRDImage1*: Dataset=Main->Autori; DataField=foto;*

Codul sursă:

Unit1.cpp – fişierul formei principale (Main)

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
#include "Unit6.h"
#include "Unit7.h"
#include "Unit8.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TMain *Main;
__fastcall TMain::TMain(TComponent* Owner): TForm(Owner){
sql="SELECT Carti.titlu, Carti.pret,Carti.pagini, Carti.data,
Editura.nume, Autori.nume, Autori.prenume, Limba.limba FROM
Autori,Carti,Editura,Limba WHERE (Carti.id_autor =
Autori.id_autor) AND (Editura.id_editura = Carti.id_editura) AND
(Limba.id_limba = Carti.id_limba)";
}
void __fastcall TMain::Carte1Click(TObject *Sender){
 Ad_carte->Show();}
```

```
void __fastcall TMain::Editura1Click(TObject *Sender){
 Ad_editura->Show();}
void __fastcall TMain::Autor1Click(TObject *Sender){
 Ad_autor->Show();}
void __fastcall TMain::Limba1Click(TObject *Sender){
 Ad_limba->Show();}
void __fastcall TMain::Vizualizare1Click(TObject *Sender){
 Admin->Show();}
void __fastcall TMain::Dupatitlu1Click(TObject *Sender){
 Cauta->Show();
 Cauta->TabbedNotebook1->PageIndex=0;
}
void __fastcall TMain::Dupalimba1Click(TObject *Sender){
 Cauta->Show();
 Cauta->TabbedNotebook1->PageIndex=1;
}
void __fastcall TMain::Dupapret1Click(TObject *Sender){
 Cauta->Show();
 Cauta->TabbedNotebook1->PageIndex=2;
}
void __fastcall TMain::Cautadupadata1Click(TObject *Sender){
 Cauta->Show();
 Cauta->TabbedNotebook1->PageIndex=3;
}
void __fastcall TMain::Toatecartile1Click(TObject *Sender){
 Query1->Close(); Query1->SQL->Text=sql;
 Query1->Open();Raport->Lista_carti->Preview();
}
void __fastcall TMain::Listadeautori1Click(TObject *Sender){
 Raport->Lista_autori->Preview();}
```

Unit2.cpp – fişierul formularului Adaugă carte

```
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TAd_carte *Ad_carte;
__fastcall TAd_carte::TAd_carte(TComponent* Owner):TForm(Owner){}
void __fastcall TAd_carte::DateTimePicker1Change(TObject *Sender){
 Main->Carti->FieldValues["data"]=DateTimePicker1->Date;
}
void __fastcall TAd_carte::Adaugaautor1Click(TObject *Sender){
 Ad_autor->Show();
}
void __fastcall TAd_carte::
Adaugaeditura1Click(TObject *Sender){
 Ad_editura->Show();
}
void __fastcall TAd_carte::Adaugalimba1Click(TObject *Sender){
 Ad_limba->Show();
}
```

Unit3.cpp – fişierul formularului Adaugă autor

```
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
TAd_autor *Ad_autor;
__fastcall TAd_autor::TAd_autor(TComponent* Owner): TForm(Owner){}
void __fastcall TAd_autor::Label3Click(TObject *Sender){
 OpenPictureDialog1->Execute();
 AnsiString s=OpenPictureDialog1->FileName;
 if(s.Length()>1){
  DBImage1->Picture->LoadFromFile(s);
  DBImage1->CopyToClipboard();
  DBImage1->PasteFromClipboard();
 }
}
void __fastcall TAd_autor::DBImage1Click(TObject *Sender){
Label3Click(Sender);
}
```

Unit4.cpp – fişierul formularului Adaugă editură

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TAd_editura *Ad_editura;
__fastcall TAd_editura::TAd_editura(TComponent* Owner):
TForm(Owner){}
```

Unit5.cpp – fişierul formularului Adaugă limba

```
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TAd_limba *Ad_limba;
__fastcall TAd_limba::TAd_limba(TComponent* Owner):TForm(Owner){}
```

Unit6.cpp – fişierul formularului Caută

```
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
```

```
#include "Unit6.h"
#include "Unit8.h"
TCauta *Cauta;
TDateTime d;int nr;
void __fastcall TCauta::Button2Click(TObject *Sender){
 AnsiString s[10];double q;
 s[0]=Main->sql+"AND (Carti.pret <:a)";
 s[1]=Main->sql+"AND (Carti.pret >:a)";
 s[2]=Main->sql+"AND (Carti.pret = :a)";
 Main->Query1->Close();
 Main->Query1->SQL->Text=s[nr];
 q=Edit2->Text.ToDouble();
 Main->Query1->ParamByName("a")->AsFloat=q;
 Main->Query1->Open();
}
void __fastcall TCauta::RadioGroup1Click(TObject *Sender){
 nr=RadioGroup1->ItemIndex;
}
void __fastcall TCauta::FormShow(TObject *Sender){
 AnsiString s;
 ComboBox1->Clear();
 Main->Limba->First();
 while(!Main->Limba->Eof){
  s= AnsiString(Main->Limba->FieldValues["Limba"]);
  ComboBox1->Items->Add(s);
  Main->Limba->Next();
 }
}
void __fastcall TCauta::Button1Click(TObject *Sender){
 Main->Query1->Close();
 Main->Query1->SQL->Text=Main->sql+"AND (Carti.titlu like :a)";
 Main->Query1->ParamByName("a")->AsString="%"+Edit1->Text+"%";
 Main->Query1->Open();
}
void __fastcall TCauta::Edit1Change(TObject *Sender){
 Button1Click(Sender);}
void __fastcall TCauta::ComboBox1Change(TObject *Sender){
 Main->Query1->Close();
 Main->Query1->SQL->Text=Main->sql+"AND (Limba.limba like :a)";
 Main->Query1->ParamByName("a")->AsString="%"+
 ComboBox1->Items->Strings[ComboBox1->ItemIndex]+"%";
 Main->Query1->Open();
}
void __fastcall TCauta::BitBtn1Click(TObject *Sender){
 AnsiString s[10];
 s[0]=Main->sql+"AND (Carti.data <=:a)";
 s[1]=Main->sql+"AND (Carti.pret >=:a)";
 s[2]=Main->sql+"AND (Carti.pret = :a)";
 Main->Query1->Close();
 Main->Query1->SQL->Text=s[nr];
 Main->Query1->ParamByName("a")->AsDateTime=d;
 Main->Query1->Open();
}
void __fastcall TCauta::DateTimePicker1Change(TObject *Sender){
```

```
 d=DateTimePicker1->DateTime;}
 void   fastcall TCauta::RadioGroup2Click(TObject *Sender){
 nr=RadioGroup2->ItemIndex;
}
void __fastcall TCauta::Tipar1Click(TObject *Sender){
 Raport->Lista_carti->Preview();
}
```

Unit7.cpp – fişierul formularului Administrare

```
#include "Unit7.h"
#include "Unit1.h"
TAdmin *Admin;
__fastcall TAdmin::TAdmin(TComponent* Owner):TForm(Owner){}
void __fastcall TAdmin::RadioGroup1Click(TObject *Sender){
 int n=RadioGroup1->ItemIndex;
 switch(n){
  case 0: DBGrid1->DataSource=Main->Carte;
  DBNavigator1->DataSource=Main->Carte;break;
  case 1: DBGrid1->DataSource=Main->Autor;
  DBNavigator1->DataSource=Main->Autor;break;
  case 2: DBGrid1->DataSource=Main->Editure;
  DBNavigator1->DataSource=Main->Editure;break;
  case 3: DBGrid1->DataSource=Main->Limbi;
  DBNavigator1->DataSource=Main->Limbi;break;
  case 4: Main->Query1->Close();
  Main->Query1->SQL->Text=Main->sql;
  Main->Query1->Open();
  DBGrid1->DataSource=Main->Interogare;
  DBNavigator1->DataSource=Main->Interogare;
  break;
 }
}
```

Unit8.cpp – fişierul pentru formularul destinat rapoartelor

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit8.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TRaport *Raport;
__fastcall TRaport::TRaport(TComponent* Owner):TForm(Owner){}
```

Probleme pentru rezolvare independentă

1. Utilitarul Database Desktop

1. Creaţi tabelul *tari*.db cu următoarele cîmpuri :

Field Name	Type	Size	Key
Nume	A	24	
Capitala	A	24	
Continent	A	24	
Suprafata	N		
Populatie	N		

a) Setaţi proprietăţile cîmpurilor astfel încît să nu fie posibilă introducerea valorilor negative în cîmpurile cu valori numerice, iar valoarea implicită a lor să fie 0.
b) Introduceţi 10 înregistrări şi salvaţi tabelul.
c) În baza tabelului *tari*.db creaţi tabelul *orase*.db cu următoarele proprietăţi

Field Name	Type	Size	Key
Nume	A	24	*
Suprafata	N		
Populatie	N		

d) Introduceţi 10 înregistrări şi salvaţi tabelul.

2. În baza tabelelor create să se creeze fişiere *sql* care vor conţine următoarele interogări:

a) interog1 va afişa din tabelul tari toate cîmpurile;
b) interog2 va afişa din tabelul tari cîmpurile Nume, Suprafata şi Populatie, unde Suprafata>2000;
c) interog3 va afişa din tabelele tari şi orase înregistrările unde cîmpul Suprafata este cuprins între 1000 şi 10000.

3. În baza tabelului *tari*.db, să se creeze fişiere *sql* care vor conţine următoarele interogări:

a) interog1 va afişa numele ţării şi densitatea populaţiei acesteia;
b) interog2 va afişa numele ţării cu densitatea populaţiei maximă;
c) interog3 va afişa numele ţării cu densitatea populaţiei minimă;

4. Creaţi tabelul *angajati*.db cu următoarele cîmpuri :

Field Name	Type	Size	Key
ID	I		*
Nume	A	20	
Prenume	A	15	
Telefon	A	15	
Data_angajarii	@		
Salariu	N		

a) Setaţi proprietăţile cîmpurilor astfel încît să nu fie posibilă introducerea valorilor negative în cîmpurile cu valori numerice, iar valoarea implicită a lor să fie 0.
b) Introduceţi 20 de înregistrări şi salvaţi tabelul.

5. Creaţi două fişiere *sql* în baza tabelului *angajati*.db, care va permite afişarea urătoarele date:

a) Cîmpurile *Nume*, *Prenume*, *Salariu*, unde *Salariu* >1500. Datele vor fi afişate descrescător după salariu;
b) Lista angajaţilor cu o vechime în muncă mai mare decît 5 ani, din data curentă.

Se vor afişa cîmpurile *ID*, *Nume*, *Prenume*, *Data_angajarii*, unde *Salariu* mai mic decît1500. Datele vor fi afişate descrescător corespunzător stagiului de muncă.

2. Aplicaţii cu Tabele

1. Elaboraţi o aplicaţie care va permite afişarea datelor din tabelele *tari*.db, *oraşe*.db, *angajati*.db. Numele tabelelor vor fi scrise într-o lista cu opţiuni (ComboBox). La selectarea uneia dintre opţiuni se va afişa pe suprafaţa formei tabelul cu datele respective. La elaborarea aplicaţiei se va utiliza cîte o singură componetă de tipul : *TTable*, *TDataSource*, *TDBGrid*.

2. Elaboraţi o aplicaţie care va afişa într-o componentă de tip TDBGrid următoarele coloane din tabelul *angajaţi*.db a aliasului BCDEMOS:

a) coloana *Nume*, fonul de culoare roşie, culoarea textului de culoare albă, lăţimea 80;
b) coloana *Prenume*, fonul de culoare galbenă, culoarea textului de culoare verde, lăţimea 50;
c) coloana *Telefon*, fonul de culoare verde, culoarea textului de culoare albă, lăţimea 60;
d) coloana *Data_angajarii*, fonul de culoare roşie, culoarea textului de culoare albă, lăţimea 50;
e) coloana *salariu*, fonul de culoare roşie, culoarea textului de culoare albă, lăţimea 50.

3. Elaboraţi o aplicaţie care în baza datelor din tabelul *angajati*.db, va permite afişarea banilor cheltuiţi pentru salarizare timp de un an. Anul se va indica prin intermediul unei componente de tip TComboBox. După selectarea anului prin intermediul unei componente de tip TLabel va fi afişată suma de bani.

4. Elaboraţi o aplicaţie care în baza datelor din tabelul *angajati*.db va permite afişarea:

a) celui mai tînăr angajat (numele, prenumele, data naşterii);
b) angajatul cu salariu maxim (numele, prenumele, salariul);
c) angajatul cu salariu minim (numele, prenumele, salariul);
d) numărul de salariaţi al căror salariu depăşeşte salariul mediu.

5. Creaţi o copie a tabelului *angajati*.db. Redenumiţi tabelul în *salariati*.db. În baza acestui tabel, elaboraţi o aplicaţie care va permite majorarea salariului cu 15%, 20%, 25% 50%. Procentul de majorare a salariului va fi indicat prin intermediul componentelor de tip TRadioGroup. Se va permite doar o singură majorare.

6. Creaţi o copie a tabelului *angajati*.db. Redenumiţi tabelul în *salariati*.db. În baza acestui tabel, elaboraţi o aplicaţie care va permite excluderea a maximum 15 angajaţi. Se vor exclude angajaţii cu cel mai mare salariu. Numărul de angajaţi care urmează a fi exclus va fi indicat prin intermediul unei componente de tip TComboBox. Se va permite excluderea repetată doar dacă numărul total de angajaţi ce au fost excluşi sau urmează a fi excluşi nu este mai mic ca 15.

7. Creaţi o copie a tabelului *angajati*.db. Redenumiţi tabelul în *salariati*.db. În baza acestui tabel elaboraţi o aplicaţie care va permite adăugarea de noi angajaţi. Pe suprafaţa formei se va afişa:

a) numărul total de angajaţi;
b) numele şi salariul angajatului cu cel mai mare salariu;
c) numele şi salariul angajatului cu cel mai mic salariu;
d) salariul mediu caracteristic tuturor angajaţilor.

8 Creaţi tabelul *salariati*.db, în baza tabelului *angajati*.db utilizînd utilitarul *Database Desktop*. Elaboraţi o aplicaţie prin intemediul căreia din tabelul *angajati*.db vor fi copiaţi în tabelul *salariati*.db 5 angajaţi cu cel mai mare salariu şi 5 angajaţi cu cel mai mic salariu.

9. Elaboraţi o aplicaţie care în baza datelor din tabelul *angajati*.db va permite căutarea unui salariat după nume. Numele va fi introdus de către utilizator prin

intermediul unei componente de tip TEdit. Dacă în tabel va exista o persoană cu numele introdus se va afişa un mesaj corespunzător. Utilizatorului i se va oferi posibilitatea de a modifica propriul nume.

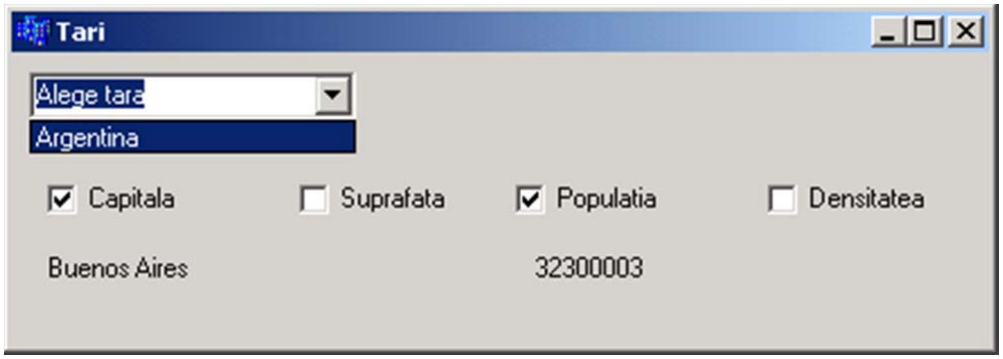
10. Să se creeze tabelul *studenti*.db cu cîmpurile *nume*, *prenume*, *grupa*, *anul*, *media*, *bursa*. De completat toate cîmpurile cu date cu excepţia Cîmpului *bursa*. Se vor introduce cel puţin 20 de înregistrări. Prin intermediul cutiilor de editare de tip *TEdit* se vor introduce numărul maxim de studenţi care pot avea bursă de gradul 1 (500 lei), 2 (400 lei) şi 3 (300 lei) cît şi media minimă a notelor pentru fiecare categorie. Prin intermediul butonului *bursa* se va completa Cîmpul bursa corespunzător.

11. Utilizînd tabelul *studenti*.db creat în exemplul 10, Creaţi tabelul *bursieri*.db şi prin intermediul butonul *completare*, să se completeze tabelul bursieri cu datele corespunzătoare.

12. Utilizînd tabelul *studenti*.db creat în exemplul 10, Creaţi tabelul *grupa_mea*.db în care se vor scrie studenţii grupei din care sunteţi. Scrierea datelor se va efectua prin intermediul butonului *completare*.

13. Utilizînd tabelul *studenti*.db creat în exemplul 10, Creaţi tabelul *stud_18*.db în care se vor scrie toţi studenţi cu vîrsta de 18 ani. Scrierea datelor se va efectua prin intermediul butonului *completare*.

14. Elaboraţi o aplicaţie prin intermediul căreia se va crea un formular de vizualizare a datelor tabelului *tari*.db. Se recomandă ca formularul să fie de următoarea formă:



15. Creaţi o aplicaţie prin intermediul căreia se va oferi posibilitatea utilizatorului de a vizualiza toate datele din tabelul *tari*.db, cît şi efectuarea operaţiilor de adăugare, eliminare şi modificare a unei înregistrări (formularul nu va conţine componente de tip TDBGrid). Vor fi create două diagrame (una dreptunghiulară, iar

alta circulară). Diagramele vor fi create în baza datelor din cîmpurile *Suprafata* şi *Populatie*.

16. Elaboraţi o aplicaţie *Ţară-Capitală*. Într-o componentă de tip *TComboBox* se vor scrie numele ţărilor din tabelul *tari*.db. Pe suprafaţa formei va fi plasată o componentă de tip TRadioGroup. În cadrul acestei componente vor fi scrise 4 nume de oraşe (dintre care unul este capitala ţării selectate). La efectuarea unui click pe una dintre opţiuni va fi afişat un mesaj cu indicarea corectitudinii răspunsului. Verificarea răspunsului se va efectua în baza datelor din tabelul *tari*.db.

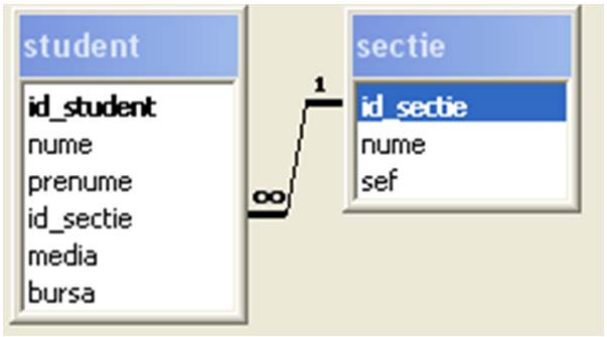
17. Elaboraţi o aplicaţie care va permite înregistrarea participanţilor la concursul de Informatică. La concurs participă concurenţi din Republica Moldova, România şi Ucraina. Din Republica Moldova vin participanţi din oraşele: Chişinău, Bălţi, Orhei, Soroca, Cahul, Edineţ. Din România vin participanţi din oraşele: Iaşi, Vaslui, Suceava, Botoşani, Galaţi. Din Ucraina vin participanţi din oraşele: Odesa, Donetk, Herson, Hotin. Datele despre participanţi se vor salva în tabelul *concurenti*.db. Tabelul va conţine următoarele cîmpuri:

Nume cimp	*tipul*	*Despre fiecare cîmp*
nume	*String*	*Utilizatorul va scrie numele (DBEdit)*
prenume	*String*	*Utilizatorul va scrie prenumele (DBEdit)*
data nasterii	*Data*	*Utilizatorul va alege data naşterii (DateTimePicker)*
tara	*String*	*Utilizatorul va alege ţara (DBListBox)*
orasul	*String*	*Utilizatorul va alege ţara (DBComboBox)*

Datele despre participanţi vor fi afişate prin intermediul componentei de tipul TDBGrid, iar prin intermediul unui meniu se va afişa:

a) numărul total de participanţi;
b) numărul de participanţi pentru fiecare ţară.

18. Elaboraţi un proiect prin intermediul căruia se va efectua o evidenţă a studenţilor în baza următoarei relaţii:



Pentru acest proiect vor fi elaborate următoarele formulare:

a) ***Formular de adăugare*** – prin intermediul acestuia se vor introduce datele despre studenţi în baza relaţiei de mai sus. Cîmpul *bursa* se va completa automat la introducerea valorilor în Cîmpul media în modul următor:
   - Dacă *media* ≤7.5 în Cîmpul bursa se va scrie valoarea 0;
   - Dacă 7.5 < *media* ≤8.5 în Cîmpul bursa se va scrie valoarea 500;
   - Dacă 8.5 < *media* ≤9.5 în Cîmpul bursa se va scrie valoarea 800;
   - Dacă 9.5 < *media* în Cîmpul bursa se va scrie valoarea 1000.

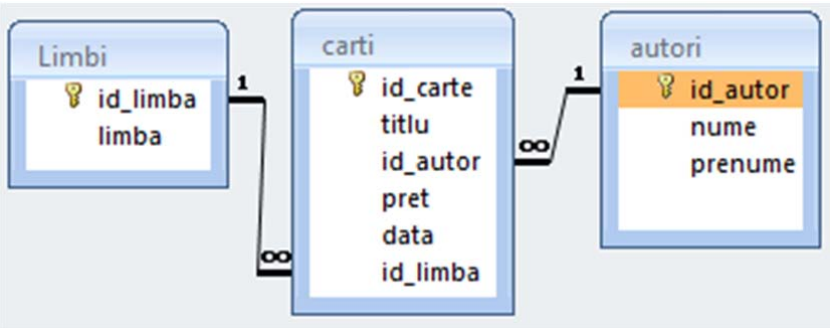
b) ***Formular de vizualizare*** (acest formular nu va conţine componente de tipul TDBGrid) – prin intermediul acestuia se vor afişa:
   - Datele despre toţi studenţii;
   - Numărul de studenţi bursieri şi suma de bani ce necesită a fi achitată pentru bursa acestora;
   - Cel mai bun student;
   - Numărul de studenţi cu media mai mică ca 5;
   - Afişarea numărului de studenţi pentru fiecare secţie.

c) ***Formular de excludere*** – prin intermediul acestuia se va permite:
   - Excluderea unui student;
   - Excluderea unei secţii (excluderea unei secţii, presupune excluderea tuturor studenţilor din secţia respectivă).

d) ***Formular pentru efectuarea de operaţii*** – prin intermediul acestuia se va permite:
   - Modificarea datelor personale despre student (nume, prenume, secţia, media. Modificarea mediei presupune şi modificarea Cîmpului bursa, în baza punctului a);
   - Modificarea datelor tabelului sectie;
   - Transferarea unui student de la o secţie la alta.

În cadrul proiectului vor fi introduse minim 15 înregistrări.

19. Elaboraţi un proiect prin intermediul căruia se va efectua o evidenţă a cărţilor în baza următoarei relaţii:

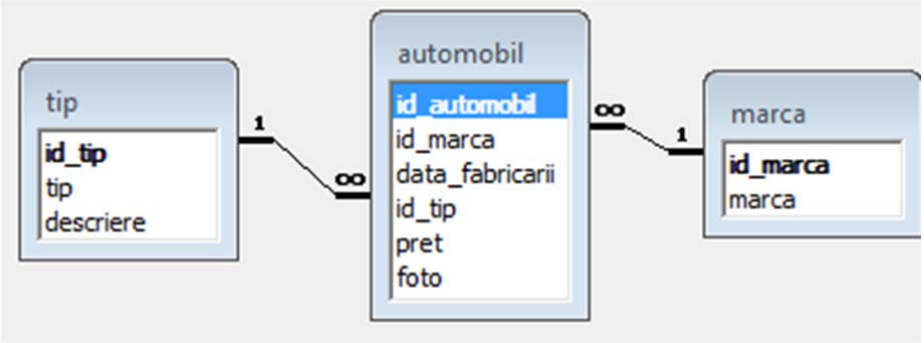


Pentru acest proiect vor fi elaborate următoarele formulare:

a) ***Formular de adăugare*** – prin intermediul acestuia se vor introduce datele despre cărţi în baza relaţiei de mai sus.
b) ***Formular de vizualizare*** (acest formular nu vor conţine componente de tipul TDBGrid) – prin intermediul acestuia se vor afişa:
   — Datele despre toate cărţile, despre toţi autorii;
   — Autorul cu cele mai multe cărţi;
   — Cartea cea mai scumpă;
   — Numărul de cărţi cu o vechime mai mare de 10 ani.
c) ***Formular de excludere*** – prin intermediul acestuia se va permite:
   — Excluderea unei cărţi;
   — Excluderea unui autor (excluderea unui autor presupune excluderea tuturor cărţilor unde figurează autorul dat);
   — Excluderea unei limbi (excluderea unei limbi presupune excluderea tuturor cărţilor unde figurează limba dată).
d) ***Formular pentru efectuarea de operaţii*** – prin intermediul acestuia se va permite:
   — Modificarea datelor despre o carte;
   — Modificarea datelor unui autor;
   — Modificarea limbii de ediţie a unei cărţi.

În cadrul proiectului vor fi introduse minim 15 înregistrări.

20. Elaboraţi un proiect prin intermediul căruia se va efectua o evidenţă a automobilelor în baza următoarei relaţii:



Pentru acest proiect vor fi elaborate următoarele formulare:

a) ***Formular de adăugare*** – prin intermediul acestuia se vor introduce datele despre automobile în baza relaţiei de mai sus.
b) ***Formular de vizualizare*** (aceste formulare nu vor conţine componente de tipul TDBGrid) – prin intermediul acestuia se vor afişa:
   — Datele despre toate automobilele;
   — Automobilul cel mai scump;
   — Numărul de automobile cu o vechime nu mai mare de 5 ani.

c) ***Formular de excludere*** – prin intermediul acestuia se va permite:

— Excluderea unui automobil;

— Excluderea unei mărci (excluderea unei mărci presupune excluderea tuturor automobilelor de marca respectivă);

— Excluderea unui tip (excluderea unui tip presupune excluderea tuturor automobilelor de tipul respectiv).

d) ***Formular pentru efectuarea de operaţii*** – prin intermediul acestuia se va permite:

— Modificarea datelor despre un automobil;

— Modificarea mărcii unui automobil;

— Modificarea tipului.

În cadrul proiectului vor fi introduse minim 15 înregistrări.

3. Aplicaţii cu Interogări

1. Elaboraţi o aplicaţie cu interogări în baza tabelului *tari*.db. Se vor afişa:

a) Toate cîmpurile;
b) Cîmpurile *Nume*, *Suprafata* şi *Populatie*, unde *Suprafata*>2000;
c) Înregistrările unde densitatea populaţiei este mai mare decît 100.

2. Elaboraţi o aplicaţie cu interogări în baza tabelului *angajati*.db, care va permite afişarea următoarelor date:

a) Cîmpurile *Nume*, *Prenume*, *Salariu*, unde *Salariu* >2500. Datele să fie aranjate după Cîmpul *Nume*;
b) Cîmpurile *Nume*, *Prenume*, *Salariu*, unde *Salariu* <2500. Datele să fie aranjate după Cîmpul *Salariu*;
c) Cîmpurile *Nume*, *Prenume*, *Salariu*, unde numele conţine unul dintre caracterele *'i'* sau *'a'*.

Într-o componentă *RadioGroup* vor fi afişate aceste trei opţiuni. La efectuarea unui click pe una dintre opţiuni, informaţia se va afişa într-o componentă *DBGrid*.

3. Elaboraţi o aplicaţie prin intermediul căreia vor fi afişate numai înregistrările tabelului *tari*.db, unde numele ţării începe cu /conţine un caracter selectat. Caracterul va fi selectat prin intermediul unei componente de tip TComboBox, care va conţine toate caracterele alfabetului englez.

4. Elaboraţi o aplicaţie prin intermediul căreia se va permite alegerea coloanelor tabelului *tari*.db. Pentru indicarea acestora se va utiliza cîte un buton de tip TCheckBox. Dacă acesta va fi bifat, atunci datele Cîmpului corespunzător se vor

afişa, în caz contrar nu se vor afişa. Datele vor fi afişate prin intermediul unei componente de tip DBGrid şi sortate după ultimul cîmp selectat.

5. Elaboraţi o aplicaţie în baza tabelului *angajati*.db, prin intermediul căreia se vor afişa:

a) Înregistrările unde Cîmpul *Salariu* $\in [a,b]$;
b) Salariul maxim pentru înregistrările cu cîmpul *Salariu* $\in [a,b]$;
c) Salariul minim pentru înregistrările cu cîmpul *Salariu* $\in [a,b]$;
d) Salariul mediu pentru înregistrările cu cîmpul *Salariu* $\in [a,b]$;
e) Numărul de înregistrări cu cîmpul *Salariu* $\in [a,b]$.

Valorile *a* şi *b* vor fi introduse de la tastatură prin intermediul a două cutii de editare.

6. Elaboraţi o aplicaţie în baza tabelului *angajati*.db, prin intermediul căreia se vor afişa:

a) Înregistrările pentru care cîmpul *Salariu* are valoare maximă;
b) Înregistrările pentru care cîmpul *Salariu* are valoare minimă;
c) Înregistrările pentru care cîmpul *Salariu* $\in [salariu\ mediu-a,\ salariu\ mediu+a]$;
d) Numărul de înregistrări, pentru care cîmpul *Salariu* $\in [salariu\ mediu-a,\ salariu\ mediu+a]$, iar cîmpul *Nume* conţine un caracter selectat prin intermediul unei componente de tip TComboBox (itemii căreia sunt toate caracterele alfabetului englez);

Parametrul *a* va fi citit de la tastatură prin intermediul unei cutii de editare.

7. Creaţi o copie a tabelului *angajati*.db. Redenumiţi tabelul în *salariati*.db. În baza acestui tabel elaboraţi o aplicaţie prin intermediul căreia:

a) vor fi majorate valorile cîmpului *salariu* cu 10%, 25%, 35%, procentul va fi ales prin intermediul unei componente de tip TRadioGroup;
b) vor fi majorate valorile cîmpului *salariu* cu 30% pentru înregistrările unde cîmpul *salariu* are o valoare mai mică decît valoarea medie;
c) vor fi reduse valorile cîmpului *salariu* cu 15% pentru înregistrările unde cîmpul *salariu* are o valoare mai mare ca valoare medie;
d) se va dubla valoarea cîmpului *salariu* de valoare minimă;
e) se va reduce valoarea cîmpului *salariu* de valoare maximă cu 50%;
f) se va exclude înregistrarea unde Cîmpul *salariu* are valoare maximă;
g) se va exclude înregistrarea unde Cîmpul *salariu* are valoare minimă.

Înainte şi după fiecare operaţie vor fi afişate toate înregistrările tabelului *salariati*.db.

8. Creaţi tabelul *salariati*.db cu următoarea structură:

	Field Name	Type	Size	Key
1	Nume	A	20	
2	Prenume	A	15	
3	Data_ang	@		
4	Salariu	N		

Elaboraţi o aplicaţie cu ajutorul căreia se vor insera înregistrări în tabelul *angajati*. Se va oferi utilizatorului posibilitatea de a selecta cîmpurile în care se vor insera date (prin intermediul a 4 componente de tip TCheckBox). Pe suprafaţa formei va fi plasat un buton de comandă cu titlul *Exclude*, la acţionarea acestuia vor fi excluse toate înregistrările din tabelul *salariati*.db.

9. Creaţi o copie a tabelului *angajati*.db. Redenumiţi tabelul în *salariati*.db. În baza acestor tabele elaboraţi o aplicaţie prin intermediul căreia vor fi efectuate următoarele operaţii:

1. excluderea tuturor datelor din tabelul salariati.db;
2. inserarea înregistrărilor din tabelul *angajati*.db în tabelul *salariati*.db unde:
   a). valoarea cîmpului *Salariu* are valoare maximă;
   b). valoarea cîmpului *Salariu* are valoare minimă;
   c). valoarea cîmpului *Salariu* $\in$[salariu mediu-a, salariu mediu+a], parametrul a se citeşte de la tastatură prin intermediul unei cutii de editare;
   d). valoarea cîmpului *Nume* conţine caracterele x, y (x şi y sunt caractere ale alfabetului latin), valorile acestora se citesc de la tastatură;
   e). valoarea cîmpului *Data_angajarii* este maximă;
   f). valoarea cîmpului *Data_angajarii* este minimă.

10. Creaţi o copie a tabelului *angajati*.db. Redenumiţi tabelul în *salariati*.db. În baza acestui tabel elaboraţi o aplicaţie prin intermediul careia se vor determina:

$$K_1=\frac{\min+\max}{avg} * count([avg-a, avg+a])$$

$$K_2=\frac{sum}{avg * \min} * count([\max-a, \max]), \text{ dacă}$$

min - valoarea minimă a cîmpului *Salariu*;
max - valoarea maximă a cîmpului *Salariu*;
a – un parametru ce se citeşte de la tastatură;
avg - valoarea medie a salariilor;
count - numărul de înregistrări cu valoarea cîmpului *Salariu* în intervalul dat;
sum - suma tuturor valorilor cîmpului *Salariu*.

După determinarea coeficienţilor K_1 şi K_2 majoraţi datele cîmpului *Salariu* după

formula:

$$Salariu = Salariu + \frac{\min + \frac{\max}{2}}{avg}$$

Calculaţi coeficienţii K_1 şi K_2 după majorare şi afişaţi rapoartele dintre vechile şi noile valori ale coeficienţilor K_1 şi K_2.

La realizarea aplicaţiei nu vor fi utilizate componente de tip TTable, numărul de parcurgeri ale tabelului *salariati*.db va fi minim.

4. Rapoarte

1. Elaboraţi o aplicaţie prin intermediul căreia va fi creat un raport pentru toate cîmpurile tabelului *tari*.db. Pagina raportului va avea următoarele dimensiunea *A4* şi orientarea *Landscape*.

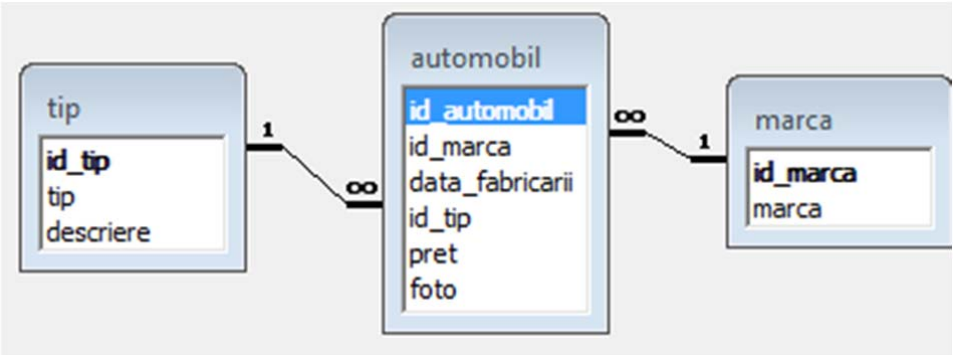
2. Elaboraţi o aplicaţie prin intermediul căreia va fi creat un raport pentru cîmpurile *Nume*, *Prenume* şi *Salariu* ale tabelului *anagajati*.db. Pe pagina raportului va fi creată o diagramă pentru cîmpurile respective.

3. Elaboraţi o aplicaţie în baza tabelei *angajati*.db, prin intermediul căreia se va crea un raport care va afişa:

a) Înregistrările unde Cîmpul *Salariu* $\in [a,b]$;
b) Înregistrările unde Cîmpul *Salariu* $\in$ *[salariu mediu-a, salariu mediu+a]*;
c) Înregistrările unde angajaţii au un stagiu de muncă nu mai mic de 5 ani.

Valorile *a* şi *b* vor fi introduse de la tastatură prin intermediul a două cutii de editare.

4. Elaboraţi o aplicaţie prin intermediul căreia vor fi elaborate rapoarte pentru fiecare dintre tabelele bazei de date cu următoarea structură:



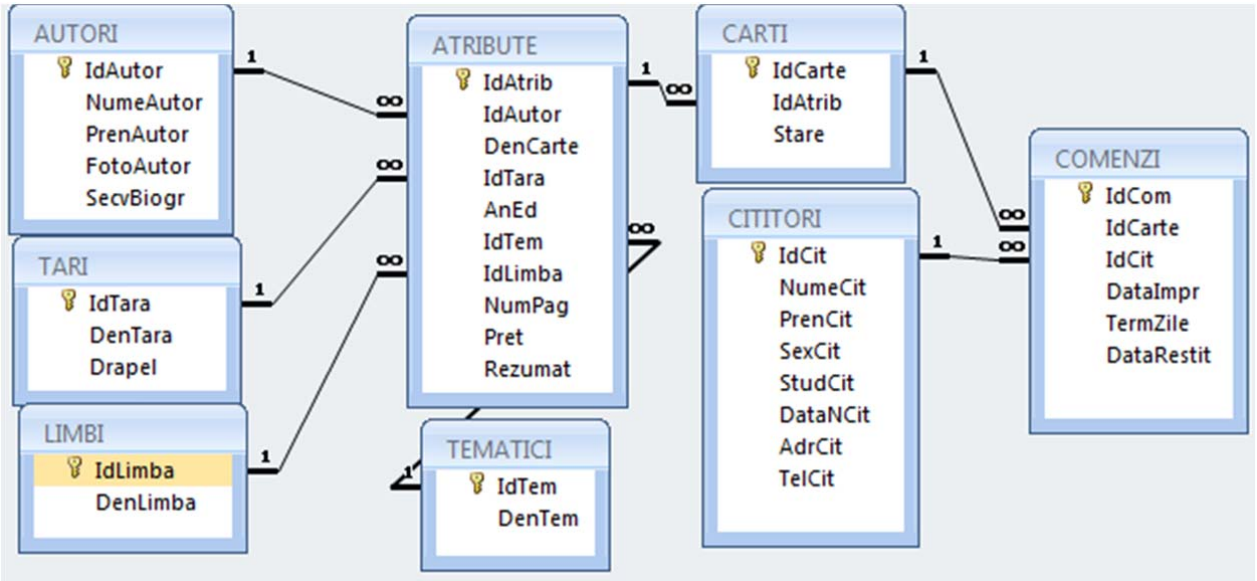
Va fi elaborat un raport prin intermediul căruia vor fi afişate cîmpurile din toate tabelele cu excepţia: automobil.foto, automobil.id_tip, automobil.id_marca, marca.id_marca, tip.id_tip.

5. Elaboraţi o aplicaţie prin intermediul căreia va fi creat un raport care va conţine orarul lecţiilor pentru săptămîna curentă. În orar se va specifica: ziua, data, lecţia, profesorul, sala de studii.

5. Tehnologia ADO

Microsoft ADO (ActiveX Data Objects) asigură acces aplicaţiilor client la date. Pentru implementarea modelului ADO Borland C++ Builder oferă paleta de componente ADO, cu ajutorul căreia se poate realiza conectarea la sursele de date suportate de ADO.

Considerăm baza de date *Biblioteca*.mdb salvată în mapa *My Documents,* cu următoarea structură:



1. Se va oferi posibilitatea de a afişa informaţia din fiecare tabel al bazei de date. La elaborarea aplicaţiei se va utiliza cîte o singură componentă de tipul: *TADOTable*, *TDataSource*, *TDBGrid*.

2. Elaboraţi o aplicaţie care va permite afişarea următoarelor informaţii din baza de date:

a) va afişa informaţia din tabelul *autori*;
b) va afişa informaţia din tabelul *cititori*;
c) din tabelul *atribute* va afişa datele următoarelor cîmpuri: *DenCarte*, *AnEd*, *NumPag*;
d) din tabelele *atribute*, *tari* va afişa datele din cîmpurile: *DenCarte*, *DenTara*, *pret*, *AnEd*;
e) va determina numărul cititorilor cu studii : *medii*, *super*, *medsp*; din tabelele

atribute, *limbi*, *tematici* va afişa datele din cîmpurile : *DenCarte*, *DenTem*, *DenLimba*, *pret*;

f) va determina numărul cărţilor cu preţul mai mic ca 50, numărul cărţilor cu preţul între 50 şi 100, numărul cărţilor cu preţul mai mare ca 100;
g) într-o componentă *ComboBox* se vor introduce numele fiecărui autor. La efectuarea unui click pe numele unui autor se vor afişa cărţile în care figurează autorul dat şi anume cîmpurile: *DenCarte* şi *pret*;
h) într-o componentă *ComboBox* se vor introduce toate datele din Cîmpul *DenLimba* din tabelul *limbi*. Alegerea uneia dintre opţiuni va afişa datele despre cărţile cu caracteristica aleasă şi anume cîmpurile: *DenCarte* şi *AnEd*.

3. Elaboraţi o aplicaţie care va mări preţul unei cărţi în modul următor: dacă numărul de pagini este mai mic ca 100 atunci preţul cărţii se va majora cu 25%, dacă numărul de pagini este mai mare sau egal cu 100, atunci preţul acesteia se va majora cu 15 %. Elaboraţi un raport în care se va prezenta informaţia despre cîmpurile *DenCarte*, *NumeAutor*, *PrenumeAutor*, *NumPag*, *pret*. Restabiliţi preţurile.

4. Elaboraţi o aplicaţie prin intermediul căreia vor fi afişate următoarele rapoarte:

a) tabelul autori;
b) tabelul tari;
c) tabelul cititori;
d) cîmpurile: DenCarte, DenTem, DenLimba, NumeAutor.

5. Elaboraţi o aplicaţie care ar permite efectuarea operaţiei de căutare, pentru următoarele cazuri:

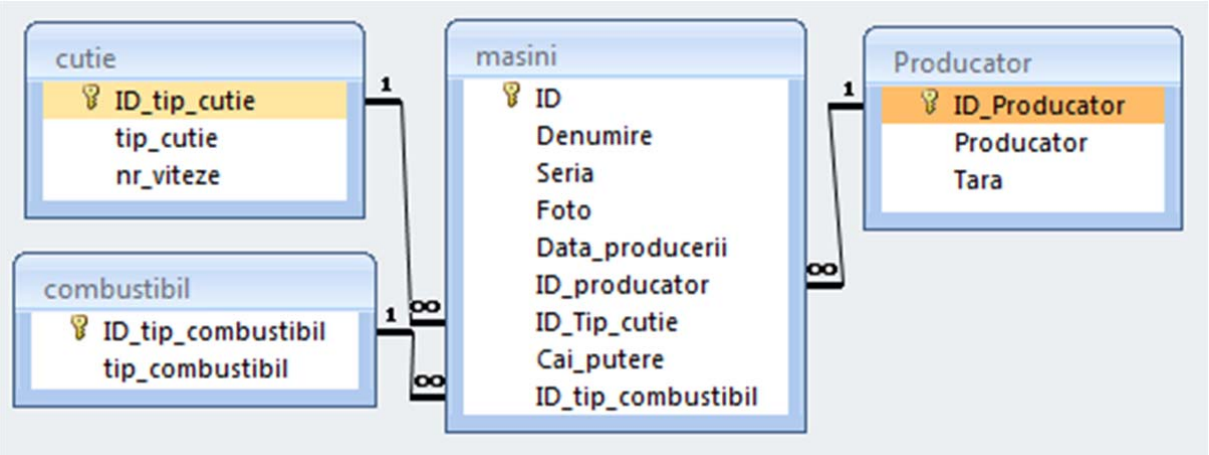
a) afişarea cărţilor după numele şi prenumele autorului;
b) afişarea cărţilor după ţara în care a fost editată;
c) afişarea cărţilor după tematică;
d) afişarea cărţilor după preţ;
e) afişarea cărţilor după starea acestora.

6. Elaboraţi o aplicaţie care la selectarea numelui şi prenumelui cititorului va afişa cartea împrumutată şi data cînd aceasta va fi restituită.

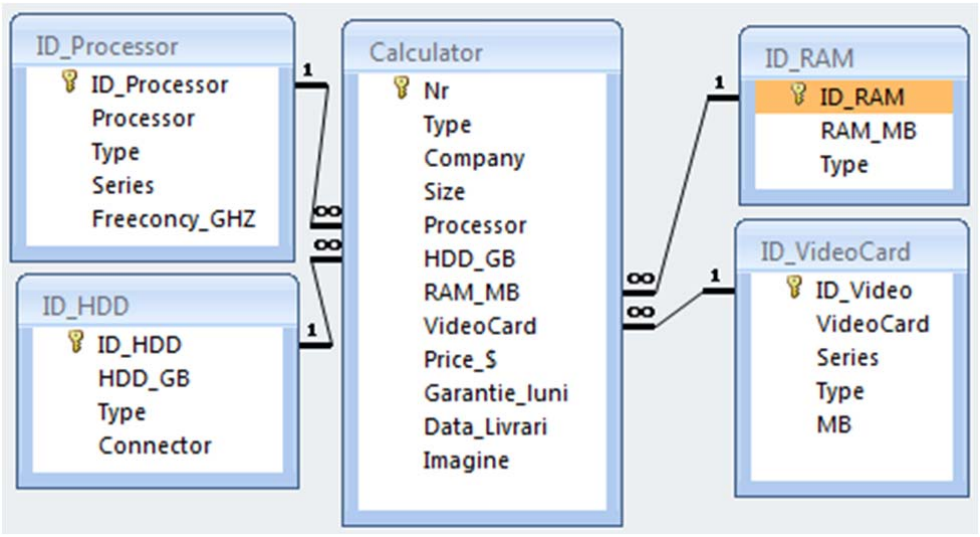
6. Proiecte pentru gestiunea bazelor de date

Pentru următoarele cazuri elaboraţi un proiect care va permite prelucrarea bazei de date:

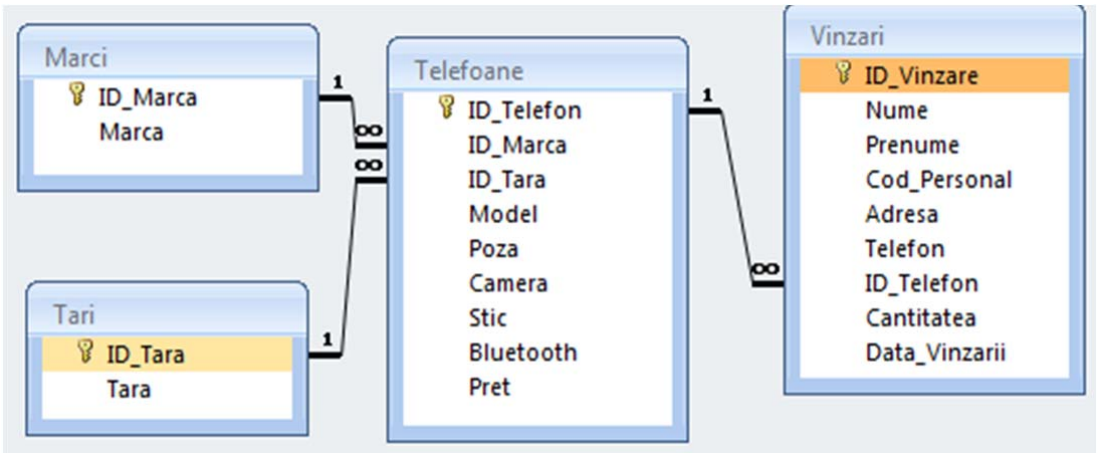
1. Depozit de automobile



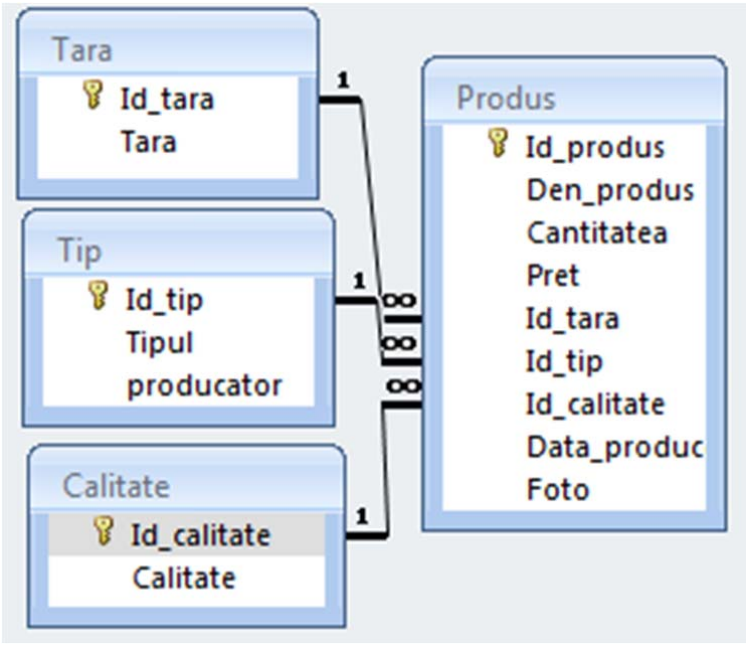
2. Magazin de calculatoare



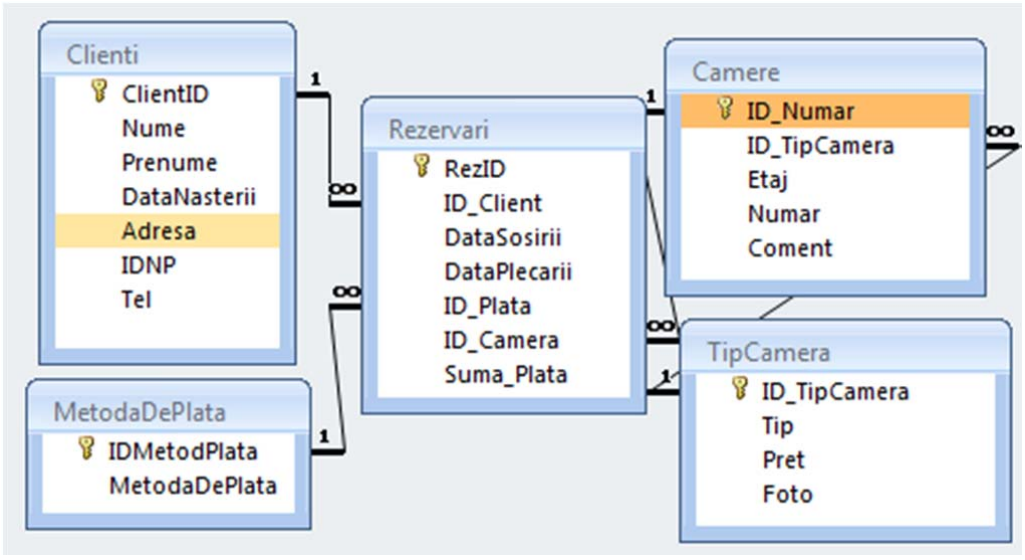
3. Salon de telefoane mobile



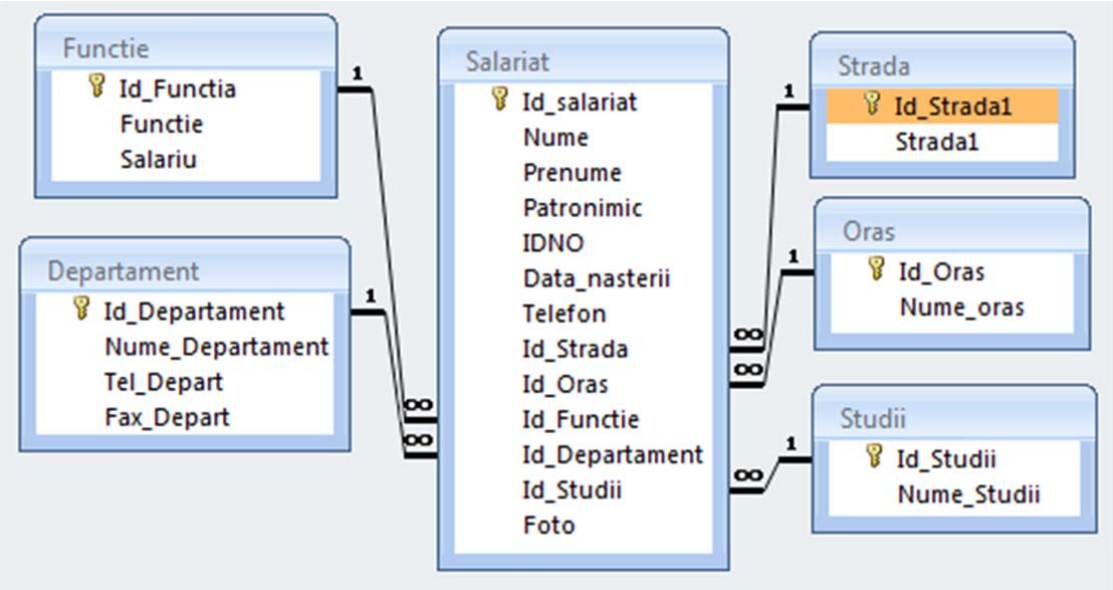
4. Depozit de produse



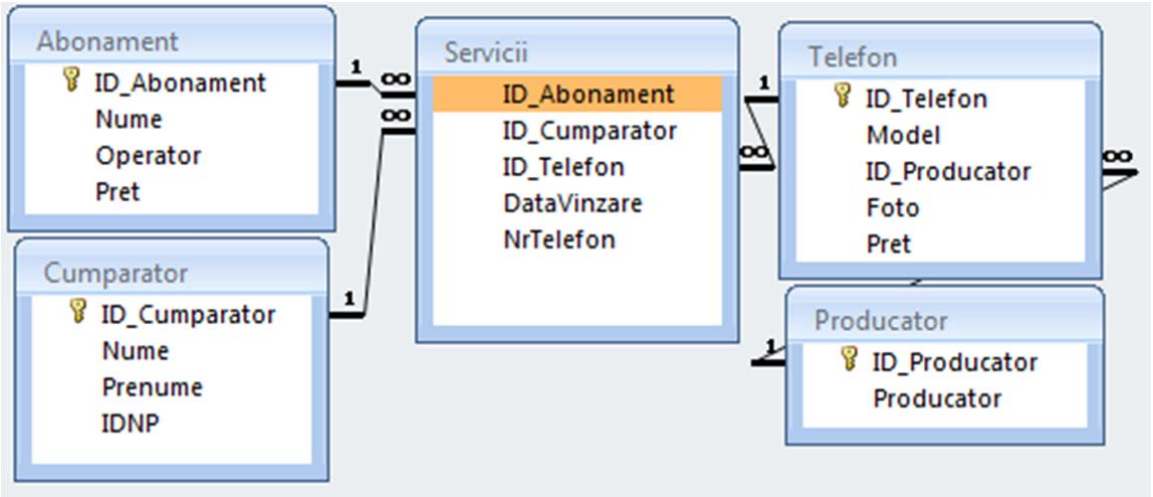
5. Rezervări la hotel



6. Angajaţi



7. Servicii de telefonie mobilă



Cerinţe faţă de proiect:

a) Formularul ferestrei principale;
b) Formulare de adăugare (pentru fiecare tabel cîte un formular);
c) Formulare pentru căutarea datelor (6 căutări);
d) Vizualizarea tuturor înregistrărilor şi vizualizarea datelor pentru fiecare tabel;
e) Crearea rapoartelor (minim 5 rapoarte);
f) Oferirea unei interfeţe simple şi comode pentru utilizatorii BD;
g) Introducerea a minim 30 de înregistrări cu date corecte;
h) * Selectarea bazei de date în timpul execuţiei.

Bibliografie

1. Braicov Andrei „Borland Delphi” Ghid de Iniţiere. Tipografia centrală 2009
2. Braicov Andrei, Gîncu Silviu „C++ Builder” Ghid de Iniţiere. Tipografia centrală 2009
3. Culea, George. Găbureanu, Cătălin. (2007) „Programarea orientată pe obiecte - Note de curs – îndrumar de laborator”, Bacău, Editura: Alma Mater
4. Grady Booch. “Object-Oriented Design with Applications”. Benjamin/Cummings, Redwood City, California, 2nd edition, 1994
5. Mihai Oltean, Crina Groşan, „Programare în C++ Builder”. Editura albastră, Cluj-Napoca 2008
6. Muşlea Ionuţ „Iniţiere în C++ Programarea orientată pe obiecte”. Cluj-Napoca 1993
7. Somnea, D., Turturea, D., Iniţiere în C++ - Programarea orientată pe obiecte, Editura Tehnică, Bucureşti, 1993
8. Stroustrup Bjarne „The C++ Programming Language”— 3rd. ed., Copyright © 1997 by AT&T
9. Spircu, C., Lopătan, I., „POO-Analiza, proiectarea şi programarea orientate spre obiecte”, Editura Teora, Bucureşti, 1996
10. Vsevolod Arnaut, Vasile Putină, Ion Andrieş „Programarea orientată pe obiecte în baza limbajului C++”. CEP USM 2009
11. Zmaranda, Rodica Doina, “Elemente de programare orientată pe obiecte în limbajul C++”, Oradea Editura Universităţii din Oradea, 2001
12. А. Я. Архангельский „Программирование в C++ Builder 6”. Издательство БИНОМ 2003
13. Бобровский С. И. „Технологии C++ Builder. Разработка приложений для бизнеса. Учебный курс”. СПб.: Питер, 2007
14. Культин Н. Б. „C++ Builder в задачах и примерах”. Петербург, 2005

Cuprins