

Temă

1) Numim **număr extins** un număr de forma $a + b\sqrt{p}$ unde a, b sunt numere întregi iar p este un număr prim. Să se definească clasa **NrExt**, clasă care să implementeze lucrul cu numere de forma definită anterior.

Cerințe obligatorii:

- Variabila p va fi declarată ca variabilă de clasă (statica) cu valoarea implicită 2
- Valoarea 0 va fi valoarea implicită pentru elementele clasei
- Se vor supraîncărca operatorii =, +, *, >>

2) Să se implementeze clasa **NumarRațional**.

Cerințe obligatorii:

- Elementul 0 va fi valoarea implicită pentru elementele clasei
- Se va implementa simplificarea
- Se vor supraîncărca operatorii =, +, -, *, /, <<, >>

3) Să se implementeze clasa **Matrice** reprezentată sub forma unui tablou unidimensional alocat dinamic. Să se supraîncarce operatorii + (suma), * (produs).

4) Să se implementeze clasa **MatriceRara**. Pentru detalii click [aici](#).

5) Să se implementeze clasa **Coordonate3D**, implementând metode pentru translație, rotație, calculul produsului scalar, calculul produsului vectorial si supraîncărcând operatorii +, -, =.

6) Să se proiecteze și să se implementeze clasa **EcGr2** care va avea elemente de forma ax^2+bx+c în care numerele a,b,c (coeficienții) au valori fixate și sunt din clasa Complex, iar variabila x poate lua diferite valori. Se vor respecta următoarele cerințe:

- Clasa va avea un constructor implicit și un constructor de copiere
- Se vor declara funcții membru pentru citirea/scrierea coeficienților
- Se va declara o funcție membru pentru evaluarea expresiei pentru un x dat
- Se va declara o funcție pentru determinarea rădăcinilor
- Se vor supraîncărca operatori

7) Să se implementeze clasa **BigDecimal** care va permite efectuarea operațiilor cu numere mari:

```
class BigDecimal {
    char *cifre;
    int len;
public:
    BigDecimal(char* a, int lungime);
    BigDecimal(BigDecimal&);
```

```

        BigDecimal(long number);
        ~BigDecimal();
        friend istream& operator >> (istream&, BigDecimal&);
        friend ostream& operator<< (ostream&, BigDecimal&);
        BigDecimal operator+ (BigDecimal&);
        int operator < (BigDecimal&);
};

```

Să se implementeze și numerele mari cu semn și să se supraîncarce în plus operatorii -, *, /.

8) Să se implementeze clasa **String** ce reprezintă siruri de caractere și operații cu acestea:

```

class String {
    char *p;
    int len;
public:
    String(char* mp);
    String(String& s);
    ~String();
    String& operator= (String& s);
    String operator+ (String& s);
    friend ostream& operator << (ostream& o, String& s);
    String operator- (String& s); //elimina toate aparitiile sirului s.
};

```

9) Folosind drept model clasa *ListaStudenti* din laboratorul 2, să se proiecteze și să se implementeze clasa **Multime** care să permită efectuarea operațiilor specifice multimediei: adaugare/eliminare elemente, intersecție, reuniune, diferență.

10) Să se scrie un program care să ajute la repartizarea lucrărilor de licență știind următoarele:

- o lucrare de licență este caracterizată de: *titlu, nivel de dificultate, domeniu* (fiecare lucrare este încadrată într-un domeniu);
 - un profesor (caracterizat prin *nume și prenume*) propune mai multe lucrări de licență, pentru fiecare lucrare propusă având o condiție asupra *punctajului minim* al studentului la domeniul în care este încadrată lucrarea;
 - un student este caracterizat de: *nume și prenume, punctaje* (fiecare student are un punctaj pentru fiecare domeniu);
 - pentru lucrarea de licență un student alege un domeniu și o listă de lucrări preferate.
- Datele de intrare vor fi citite din fișiere.

11) **Sudoku** este un joc de logică format dintr-un puzzle de numere pe o foaie de 9 pătrățele, fiecare cu câte 9 alte pătrățele în interior. În aceste casuțe, la pornirea jocului există deja anumite numere. Scopul jocului este acela de a umple restul de casuțe goale în așa fel încât fiecare pătrat mare (de 3 x 3) să conțină numere de la 1 la 9, toate apărând doar o singură dată. Mai mult, numerele introduse într-un pătrat mare sunt legate și de cele din celelalte pătrate mari deoarece pe o linie respectiv coloană un număr nu poate să apară de două ori.

Detalii suplimentare se pot găsi la adresa: <http://ro.wikipedia.org/wiki/Sudoku>

Exemplu:

9			1					5
		5		9		2		1
8				4				
				8				
			7					
				2	6			9
2			3					6
			2			9		
		1	9		4	5	7	

Să se scrie un program care să ofere un meniu cu următoarele variante:

- Generarea unui joc nou, funcție de nivelul de dificultate specificat;
- Pentru un joc dat (citirea făcându-se dintr-un fișier) să se determine soluția și să se adauge în același fișier.

Pentru generarea de jocuri noi se pot urmări indicațiile din laboratorl 3.

12) Să se realizeze o aplicație C++ care să modeleze jocul **Hangman** (Spanzuratoarea).

Cerințe:

- se va alege un cuvânt la întâmplare dintr-o listă predefinită de cuvinte (*cuvinte.txt*) care va reprezenta soluția curentă a jocului. Tot din fișier se va prelua și numărul maxim de încercări disponibile
- jucatorul încearcă ghicirea cuvântului literă cu literă
- dacă este ghicită o literă atunci vor fi afișate toate aparițiile ei
- jocul va fi câștigat dacă jucătorul ghicește toate literele cuvântului înainte să depășească numărul maxim de încercări
- în timpul jocului se va afișata starea curenta a cuvântului care trebuie ghicit.

13) Să se exemplifice utilizarea următoarelor noțiuni:

- moștenire simplă
- moștenire multiplă
- funcții virtuale
- clase abstracte

prin realizarea unei aplicații pentru evidența agajaților într-o firmă.

Restricții:

- funcțiile angajaților sunt organizate într-un sistem ierarhic, acestea putând fi muncitor sau manager

- un manager poate fi CEO, Șef departament sau Team leader. Un muncitor nu are nici un subordonat. Un șef de departament poate avea subordonați mai mulți team leaderi.
- fiecare angajat va avea: Nume, Salariu, Functie, Departament și lista cu angajații subordonați.
- informațiile vor fi păstrate în fișiere

Aplicatia trebuie să permită următoarele acțiuni:

- afișarea informațiilor despre toți angajații
- total cheltuieli cu salariile
- introducerea unui nou angajat (angajarea unei persoane în firmă)
- modificarea salariului unui angajat

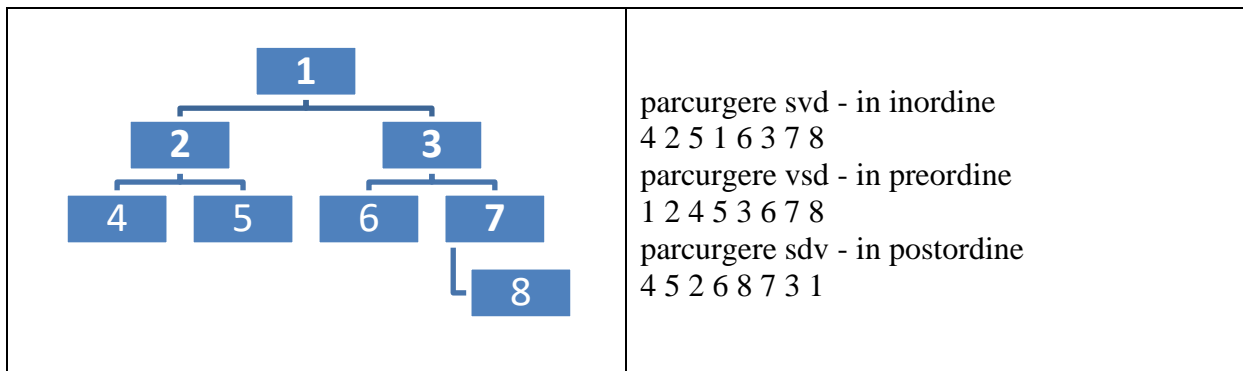
14) Să se implementeze (fără STL) clasa **ArboreBinar**.

Cerințe:

- nodurile pot fi de orice tip: numere intregi, reale, obiecte (va fi clasă template)
- clasa va conține metodele:
 - creare arbore
 - parcurgere în inordine
 - parcurgere în postordine
 - parcurgere în preordine

Metode specifice arborilor binari :

- Parcurgerea în inordine (stânga –vârf – dreapta SVD) – se parcurge mai întâi subarborele stâng, apoi vârful, apoi subarborele drept.
- Parcurgerea în preordine (varf- stanga – dreapta VSD) – se parcurge mai întâi varful, apoi subarborele stâng, apoi subarborele drept.
- Parcurgerea în postordine (stânga – dreapta – vârf SDV) – se parcurge mai întâi subarborele stâng, apoi subarborele drept și la sfârșit vârful.



15) Să se scrie un program în care să se utilizeze cel puțin trei dintre clasele următoare:

<http://www.cplusplus.com/reference/list/list/>
<http://www.cplusplus.com/reference/vector/vector/>
<http://www.cplusplus.com/reference/set/set/>
<http://www.cplusplus.com/reference/map/map/>
<http://www.cplusplus.com/reference/string/string/>
<http://www.cplusplus.com/reference/iterator/iterator/>
<http://www.cplusplus.com/reference/algorithm/>

16) Aceeași cerință ca la 14 dar cu STL.

17) După modelul clasei Matrice prezentat în cadrul laboratorului, să se rescrie clasa Polinom astfel încât să fie o clasă template (generică).

Indicație: Pentru supraîncărcarea operatorului de citire se poate folosi codul de mai jos, unde am tratat separat tipurile native:

```
template<typename T>
istream& operator>>(istream& in, Polinom<T>& P) {
    char linie[1000];
    in.getline(linie, 1000);
    int gr = -1;
    T cf[100];
    char* pt;
    pt = strtok(linie, " ");
    while (pt) {
        if (typeid(T) == typeid(char) || typeid(T) == typeid(short) ||
            typeid(T) == typeid(int) || (T) == typeid(long) ||
            typeid(T) == typeid(float) || (T) == typeid(double)) {
            cf[++gr] = atof(pt);
        }
        else { cf[++gr] = T(pt); }
        pt = strtok(NULL, " ");
    }
    P = Polinom<T>(gr, cf);
    return in;
}
```

18) Utilizând clase template să se scrie un program pentru implementarea unei liste dublu înlanțuite. După implementare să se folosească o listă pentru gestionarea unor elemente de tip Persoana: Student, StudentAngajat.

Se poate porni de la următoarele: [Nod.h](#), [Nod.cpp](#), [ListaD.h](#), [ListaD.cpp](#), [Source.cpp](#).

19) Să se definească clasa *Student* care va moșteni clasa *Persoana* apoi să se scrie un program care să citească studenți dintr-un fișier și să-i adauge într-o stivă. Pentru student acem pe lângă *nume* și *cnp* campurile: *punctaj* și *an*.

La citirea informațiilor din fișier se va avea în vedere:

- se extrag liniile care nu au formatul corespunzător
- se extrag liniile care conțin date greșite (se vor folosi excepțiile definite la clasa persoana)
- liniile cu probleme sunt afișate într-un fișier sau la consolă în forma:

```
<linie>
    <numar_linie>1<\numar_linie>
    <text_linie>Popescu Vasile,1870814336046,16o,3<\text_linie>
    <cod_eroare>3<\cod_eroare>
    <descriere_eroare>PUNCTAJ INCORECT<\descriere_eroare>
<\linie>
```

Pentru stivă se va folosi una din variantele template definite în laboratoarele anterioare.

20) Să se realizeze o interfață grafică pentru aplicația descrisă în problema 4 din laboratorul 9 (cea cu taxa de poluare). Să se implementeze citirea/scrierea coeficienților de calcul din/în fișiere.

21) Să se implementeze operațiile de import/export pentru datele de contact (în format compatibil cu cel acceptat de telefoanele mobile) pentru aplicația Agenda descrisă în laboratorul 11.

22) Să se scrie varianta C# a problemei 11 și să se realizeze o interfață pentru aplicație. Se vor utiliza metode grafice. (Indicație: se poate lua drept model aplicația Șah făcută în cadrul laboratorului).

23) Să se scrie varianta C# a problemei 12 și să se realizeze o interfață pentru aplicație. Se vor utiliza metode grafice.

24) **Codul de bare** este o reprezentare de date codificată (cifrată), destinată a fi citită pe cale optică. Codurile de bare sunt folosite în multe domenii, îndeosebi industriale. Un cod de bare are aspectul unui șir de bare negre de diverse grosimi pe un fundal alb. În general fiecare cifră sau literă se reprezintă printr-o anumită combinație de 1 sau mai multe bare. Există mai multe formate (sisteme) de coduri de bare.

În cele ce urmează definim un sistem de coduri de bare astfel:

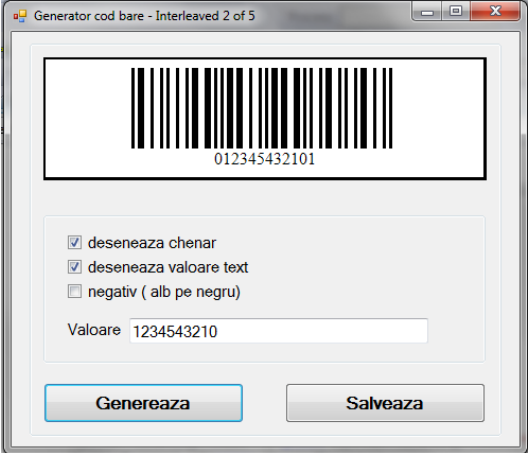
- Valorile acceptate trebuie să aibă un număr par de cifre
- Valorile sunt compuse doar din cifre (0,1,...,9)
- Fiecare cifră este reprezentată de 5 bare (3 cu grosimea l , 2 cu grosimea L -a se vedea tabelul de mai jos)
- Fiecare bară are culoarea alb sau negru dar să nu existe bare vecine de aceeași culoare
- Pentru implementarea verificării se adaugă valoarea "0" la început iar la final se adaugă o cifră (cheie) care se calculează astfel:
 - se calculează suma cifrelor de pe poziții impare
 - se calculează suma cifrelor de pe poziții pare și se înmulțește cu 3
 - se face suma celor două valori
 - cheia căutată este cel mai mic număr care trebuie adăugat valorii obținute pentru a obține un număr divizibil cu 10

Caracter	B1	B2	B3	B4	B5
0	0	0	1	1	0
1	1	0	0	0	1
2	0	1	0	0	1
3	1	1	0	0	0
4	0	0	1	0	1
5	1	0	1	0	0
6	0	1	1	0	0
7	0	0	0	1	1
8	1	0	0	1	0
9	0	1	0	1	0

0 – corespunde unei bare înguste
1 – corespunde unei bare late

Exemplu:

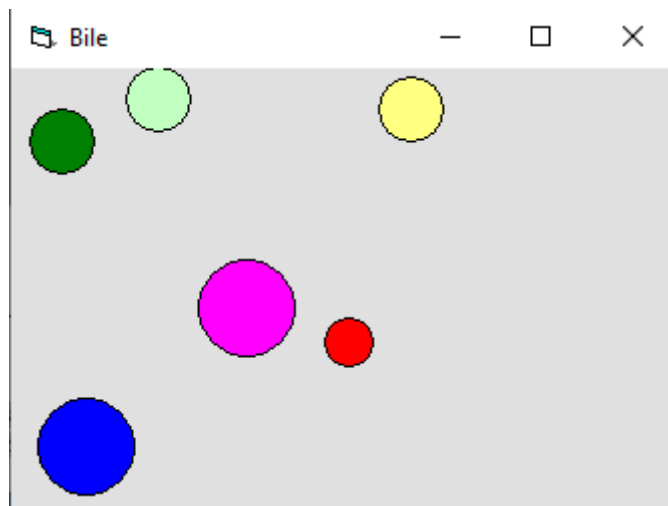
Dacă valoarea inițială este “1234543210” atunci vom adăuga la început valoarea “0” iar la final adăugăm cheia c , pe care o calculăm astfel:

$\begin{aligned}1+3+5+3+1 &= 13 \\ 2+4+4+2+0 &= 12 \\ 12*3 &= 36 \\ 13+36 &= 49 \\ c &= 50-49=1\end{aligned}$	
---	--

Valoarea de reprezentat va fi “012345432101”.

Să se definească o clasă în care să fie implementate metode pentru generarea codurilor de bare respectiv pentru citirea codului de bare dintr-o imagine.

25) Să se modeleze prin intermediul unor clase mișcarea unor bile pe o suprafață plană. Se va ține seama de coeficientul de frecare și de impulsul inițial asupra uneia dintre bile. (Se punctează și modul în care este proiectată aplicația).



26) Să se realizeze un joc de tipul "**Word Search**".

Descriere:

Într-un pătrat cu n linii și n coloane se vor genera în mod aleatoriu litere astfel încât printre acestea să se găsească p cuvinte, conform exemplului de mai jos. Cele p cuvinte se vor genera

aleatoriu conform specificațiilor utilizatorului (limbă, categorie, număr maxim de caractere, număr minim de caractere) din fereastra de configurare. Cuvintele pot fi scrise pe orizontală, pe verticală sau pe diagonale, în orice direcție.

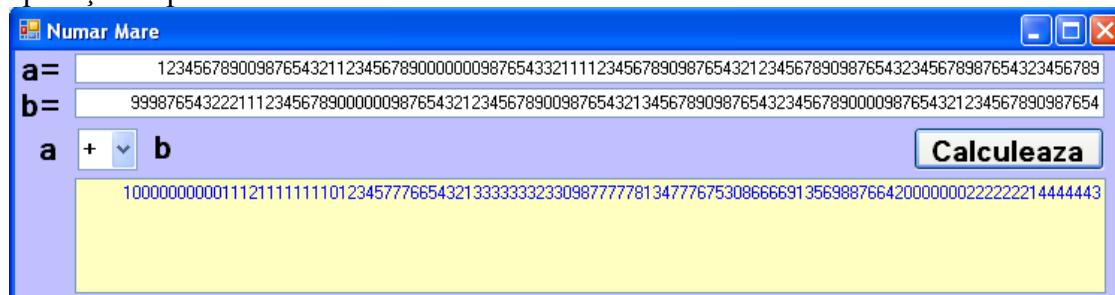
Exemplu:

E	B	K	D	K	X	S	L	A	P	D
L	R	D	B	G	L	D	F	O	S	A
E	K	A	T	T	U	J	M	I	M	F
A	X	E	M	Q	H	C	B	H	V	J
J	S	C	M	A	K	D	N	Y	B	E
T	C	E	I	O	R	P	T	R	W	Q
L	Z	S	N	M	X	G	G	S	D	P
D	Q	G	W	I	N	D	O	W	L	L
K	Y	Z	P	R	O	T	E	R	S	H
A	H	Y	W	S	D	A	B	P	P	X
S	D	F	E	R	U	I	B	N	K	Z

Exemplul anterior a fost generat pentru $n=11$, $p=3$, limba=română, categorie=oricare, numar maxim caractere=10, numar minim caractere=5.

Utilizatorului trebuie să i se afișeze lista cu toate cuvintele ce trebuie găsite. Aceste cuvinte trebuie găsite în timpul specificat (în fereastra de configurare). Un cuvânt găsit trebuie să poată fi selectat cu mouse-ul. Odată găsit, un cuvânt trebuie să rămână încercuit (conform exemplului de mai sus) și să fie tăiat din lista cuvintelor ce trebuie căutate. Din momentul începerii jocului până la final (găsirea tuturor cuvintelor sau renunțare din partea utilizatorului sau expirarea timpului) trebuie să fie afișat timpul scurs și timpul rămas. În momentul găsirii tuturor cuvintelor, utilizatorului i se va comunica timpul scurs de la începutul jocului, scorul realizat și i se va cere un nume. Se va afișa apoi o listă cu userii care au realizat cele mai bune 10 scoruri. Alături de scor se vor afișa: n , p , timpul (în secunde) și data (trebuie să apară valori de tipul: acum 10 minute sau acum 5 zile sau acum 2 săptămâni sau acum 6 luni etc). Scorul trebuie să fie direct proporțional cu n , p și invers proporțional cu timpul scurs până la găsirea tuturor cuvintelor.

27) Să se scrie varianta C# a problemei 7 și să se realizeze o interfață pentru aplicație. Interfața aplicației ar putea fi



Se vor avea în vedere următoarele:

- realizarea interfeței grafice și implementarea claselor realizate
- supraîncărcarea operatorilor: $<$, $>$, $==$, $<=$, $>=$
- supraîncărcarea operatorilor: $+$, $-$, $++$, $--$, $+=$, $-=$

- supraîncărcarea operatorilor: *, /, %
- implementarea metodei `static BigDecimal radical(BigDecimal nr);`
- implementarea unei metode pentru generarea numerelor aleatoare

28) Să se scrie o aplicație în care să se exemplifice utilizarea unor șabloane de proiectare (Design Patterns).

29) Utilizând șabloane de proiectare, să se scrie o aplicație pentru restaurante. Trebuie modelate următoarele: comenzile sunt date chelnerilor iar aceștia le transferă bucatariei unde se generează comenzile (indicație: builder). Comenzile sunt păstrate de către bucătarul șef într-o formă care să asigure că fiecare comandă este tratată o singură dată (indicație: singleton).

30) Să se creeze clasa Fiveton după modelul Singleton astfel încât să existe maximum $n=5$ instanțe, iar de la al 6-lea apel `getInstance()` încolo să returneze una din cele 5 instanțe conform algoritmului round robin). Clasa va conține și un atribut *ID* prin intermediul căruia se va ști ce instanță a fost returnată.