

## LUCRARE DE LABORATOR NR. 2

**Tema:** Constructorul – funcție de inițializare a obiectelor clasei

**Scopul lucrării:**

- Studiarea principiilor de definire și utilizare a constructorilor
- Studiarea principiilor de definire și utilizare a destructorilor
- Studiarea tipurilor de constructori

### Noțiuni de bază

Argumentarea necesității

Una din cele mai răspândite erori de programare (în orice limbaj) constă în utilizarea obiectelor fără inițializare anterioară, așa precum nu toate limbajele asigură inițializare automată. Desigur, poate fi definită funcția de inițializare și de distrugere a obiectului:

```
class Book{
    char *author;
    int year;
    int pages;
public:
    void Init(char*, int, int);
    void Destroy();
};

void Book::Init(char* a, int y, int p){
    author = new char[strlen(a)+1];
    strcpy(author,a);
    year=y;
    pages=p;
}

void Book::Destroy(){
    delete[] author;
}
```

În acest exemplu, însă, nimeni nu garantează că inițializarea va avea loc, sau că va fi eliberată memoria alocată. Alt neajuns al exemplului dat constă în pericolul scurgerii de memorie, deoarece funcția de inițializare poate fi apelată de nenumărate ori. De asemenea se poate bloca sistemul din cauza utilizării eronate a memoriei dinamice, motivul fiind apelul funcției `Destroy` fără inițializare.

Definire și utilizare

Pentru a evita această eroare, C++ asigură mecanismul de inițializare automată pentru clasele definite de utilizator – *constructorul* clasei. Iar pentru operațiile de distrugere – *destructorul* clasei.

Constructorul – este o funcție membru specială, de același nume cu numele clasei, care se apelează automat la crearea obiectului de tipul clasei. Constructorul nu returnează nici un rezultat, chiar nici *void*. Compilatorul garantează apelarea unică a constructorului pentru un singur obiect.

Destructorul – este o funcție membru specială, care se apelează automat la distrugerea obiectului. Numele unui destructor este numele clasei precedat de caracterul „~”. Compilatorul

garantează apelarea unică a destructorului pentru un singur obiect. Un destructor nu are parametri, de aceea nu poate fi supraîncărcat, și el este unic pentru o clasă.

Pentru un obiect local, destructorul se apelează când controlul programului iese din domeniul lui (se iese din blocul în care este declarat). Obiectele dinamice nu pot fi distruse automat. Distrugerea se realizează de către programator prin intermediul operatorului *delete*, care apelează, propriu zis, destructorul.

Fie un exemplu analogic cu cel precedent, utilizând constructorul și destructorul:

```
class Book{
char *author;
int year;
int pages;
public:
    Book(char*, int, int);
    ~Book();
};
Book::Book(char* a, int y, int p){
    author = new char[strlen(a)+1];
    strcpy(author,a);
    year=y;
    pages=p;
}
Book::~~Book(){
    delete[] author;
}
void main(){
    Book b("Stroustrup",2000,890);
    // Book b1; // încercarea de a crea obiectul fără apelul
                //constructorului duce la erori

    Book* ptr = new Book("Lippman",1996, 1200);

                // crearea dinamică a obiectului
    delete ptr;
                // în acest caz eliberarea memoriei se realizează de
                //către programator
}
// aici se apelează automat destructorul pentru b.
```

#### **Tipurile constructorilor**

Există patru tipuri de constructori: implicit, de copiere, de conversie a tipului și general. Această clasificare este bazată pe regulile de definire și utilizare a constructorilor.

- Constructorul implicit – constructor fără parametri, sau constructor cu toți parametrii implicați.
- Constructorul de copiere – constructor care are ca parametru referință la obiectul clasei respective. Constructorul de copiere poate avea și alți parametri care însă trebuie să fie implicați.
- Constructorul de conversie a tipului - constructorul, care transformă un tip de obiect în altul.
- Constructorul general – constructorul care nu corespunde categoriilor enumerate mai sus.

Fie exemplul:

```
class Book{
char *author;
int year;
int pages;
public:
    Book();           // constructor implicit
    Book(const Book&); // constructor de copiere
    Book(const char*); // constructor de conversie a tipului
    Book(char*, int, int); // constructor general
    ...
};
...
void main(){
    Book b("Stroustrup",1997,890); // general
    Book b1 = b, b11(b);           // de copiere
    Book b2 = "Stroustrup", b21("Bjarne");
    // de schimbare a tipului
    Book b3;                        // implicit
}
```

#### **Constructorii generați de compilator. Interacțiunea lor cu constructorii definiți de utilizator**

Prezența constructorilor nu este obligatorie. Se pot defini clase și fără constructori. În acest caz, compilatorul C++ generează în mod automat un constructor implicit și de copiere.

Constructorul implicit generat nu realizează absolut nimic, iar constructorul de copiere generat automat implică copierea bit cu bit, ceea ce nu întotdeauna este satisfăcător. În cazul definirii unui constructor de către utilizator se generează automat numai constructorul de copiere.

#### **Recomandări**

Fiecare clasă trebuie să conțină constructori. Dacă clasa conține elemente pointeri trebuie neapărat să se supraîncarce constructorul de copiere, care se utilizează pentru crearea copiilor obiectului, la apelul obiectului în funcție. Motivul supraîncărcării constructorului de copiere constă în necesitatea utilizării acțiunilor suplimentare în comparație cu algoritmul standard de copiere, deoarece se poate întâmpla ca două obiecte diferite să refere una și aceeași memorie, ceea ce poate duce la pierderea informației și la erori în sistemul de operare.

#### **Exemplu de constructori de copiere:**

```
Book::Book(const Book& b){
    author = new char[strlen(b.author)+1];
    strcpy(author, b.author);
    year = b.year;
    pages = b.pages;
}
```

Constructorii trebuie să utilizeze mecanisme numite inițializatori (lista de inițializare). Această regulă contribuie la evitarea cheltuielilor necesare pentru inițializarea câmpurilor clasei, deoarece la execuția constructorului câmpurile trebuie să fie deja inițializate, iar la execuția inițializatorilor – nu.

```
Book::Book(const Book& b): year(b.year),pages(b.pages),  
    author(new char[strlen(b.author)+1])  
{  
    strcpy(author, b.author);  
}  
// apelul funcției strcpy a rămas în afara inițializatorului.
```

În cazul utilizării tabloului de obiecte se definește în mod obligatoriu un constructor implicit, care se apelează pentru inițializarea elementelor tabloului.

### **Întrebări de control:**

1. Explicați termenul de inițializare.
2. Ce cuvinte cheie se utilizează pentru definirea constructorului și a destructorului?
3. Poate oare un constructor să returneze o valoare?
4. Pot oare să existe clase fără constructori?
5. Câți constructori poate conține o clasă? Dar destructori? De ce?
6. Enumerați tipurile de constructori?
7. Cum se apelează un constructor?
8. De ce este nevoie de definit un constructor de copiere?
9. De ce este nevoie de definit un constructor de conversie a tipului?
10. În ce cazuri se apelează constructorul de copiere? Care este sintaxa apelului?
11. Ce sunt listele de inițializare (inițializatori)?
12. În care caz se definește obligatoriu constructorul implicit?

**Sarcina  
Varianta 1**

a) Să se creeze clasa *Date* – dată cu câmpurile: zi(1-28..31), lună(1-12), an (numere întregi). Să se definească constructorii; funcțiile membru de setare a zilei, lunii și anului; funcțiile membru de returnare a zilei, lunii, anului; funcțiile de afișare: afișare tip „ 6 iunie 2004” și afișare tip „6.06.2004”. Funcțiile de setare a câmpurilor clasei trebuie să verifice corectitudinea parametrilor primiți.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *int*, numărul de rânduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.

**Varianta 2**

a) ) Să se creeze clasa *Time* – timp cu câmpurile: ore (0-23), minute (0-59), secunde (0-59). Să se definească constructorii, funcțiile membru de setare a timpului; de returnare a orei, minutelor și secundelor; funcțiile de afișare conform șablonului: „ora 17 și 18 minute și 4 secunde” și „5 p.m. 18 minute 4 secunde”. Funcțiile de setare a câmpurilor clasei trebuie să verifice corectitudinea parametrilor primiți.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *double*, numărul de rânduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.

**Varianta 3**

a) Să se creeze clasa *Stack* – stivă. Câmpurile – numărul general și numărul utilizat de elemente și pointer pentru alocarea dinamică a memoriei. Să se definească constructorii: implicit, de copiere și cu parametru – numărul necesar de elemente; funcțiile *Push* și *Pop* pentru punerea unui element pe stivă și scoaterea unui element din stivă respectiv; funcțiile *IsEmpty* și *IsFull* pentru determinarea stării stivei. Funcția *isEmpty* returnează valoarea 1 dacă stiva este vidă și zero în caz contrar, iar funcția *isFull* returnează valoarea 1 dacă stiva este plină și zero în caz contrar.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *float*, numărul de rânduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se

testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.

#### **Varianta 4**

a) Să se creeze clasa *File* – fișier, care conține informația despre numele complet al fișierului și anexa de asociere (doc – Word, psd – Photoshop, etc.), utilizând memoria dinamică, mărimea, data și timpul creării. Să se definească toți constructorii. Constructorul de schimbare a tipului are parametrul – numele de fișier. Să se definească funcțiile de redenumire a fișierului, de transfer în alt directoriu și de modificare a anexei de asociere.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *long*, numărul de rînduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.

#### **Varianta 5**

a) Să se creeze clasa *Document* – document, care conține informația despre denumirea, tema, autorul documentului utilizând memoria dinamică; numărul de pagini, data și timpul ultimei redactări. Să se definească toți constructorii. Constructorul de schimbare a tipului are ca parametru denumirea documentului. Să se definească funcțiile de modificare a temei, datei ultimei redactări ș. a.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *char*, numărul de rînduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.

#### **Varianta 6**

a) Să se creeze clasa *Image* – imagine, care conține următoarea informație: denumirea fișierului, formatul de compresie, dimensiunile imaginii, dimensiunea în octeți, compresia (în %). Să se definească toți const

ructorii. Constructorul de schimbare a tipului are ca parametru denumirea fișierului. Să se definească funcțiile de modificare a denumirii de fișier, a formatului, a dimensiunii ș. a.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *Complex*, numărul de rînduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se

testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.

#### **Varianta 7**

a) Să se creeze clasa *Queue* – coadă. Cîmpurile – numărul de elemente și un pointer pentru alocarea dinamică a memoriei. Să se definească constructorii: implicit, de copiere și cu un parametru – numărul necesar de elemente; funcțiile *add* și *get* pentru punerea unui element în coadă și pentru scoaterea unui element din coadă respectiv; funcțiile: *isEmpty*, care returnează valoarea 1 dacă coada este vidă și zero în caz contrar, și *isFull* care returnează valoarea 1 dacă coada este plină și zero în caz contrar.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *int*, numărul de rînduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.

#### **Varianta 8**

a) Să se creeze clasa *String* – șir, utilizînd memoria dinamică. Să se definească constructorii: implicit, de copiere și cu un parametru – pointer spre un șir de tip *char*. Să se definească funcțiile de atribuire a unui șir la altul, de comparație, de căutare a unui subșir, de numărare a simbolurilor ș. a.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *double*, numărul de rînduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.

#### **Varianta 9**

a) Să se creeze clasa *Discipline* - disciplină, care conține informația despre denumirea obiectului, numele profesorului, numărul de ore la disciplină și forma de susținere (examen sau colocviu). Să se utilizeze memoria dinamică pentru setarea cîmpurilor textuale. Să se definească constructorii: implicit, de copiere și cu un parametru – denumirea disciplinei. Să se definească funcțiile de schimbare: a profesorului, a numărului de ore și a formei de susținere.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *float*, numărul de rînduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se

testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.

#### **Varianta 10**

a) Să se creeze clasa *Soft* – fișier, care conține informația despre numele complet al fișierului și anexa de asociere (doc – Word, psd – Photoshop, etc.), utilizând memoria dinamică, mărimea, data și timpul creării. Să se definească toi constructorii. Constructorul de schimbare a tipului are parametrul – numele de fișier. Să se definească funcțiile de redenumire a fișierului, de transfer în alt directoriu și de modificare a anexei de asociere.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *long*, numărul de rînduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.

#### **Varianta 11**

a) Să se creeze clasa *Group* – grupa, care conține informația despre codul grupei, specialitatea, numărul de studenți, curatorul grupei și anul de studii. Să se utilizeze memoria dinamică pentru setarea câmpurilor textuale. Să se definească: toate tipurile de constructori; funcțiile de schimbare a curatorului și a numărului de studenți. Să se definească funcția de setare a anului de studii în așa fel, încît valoarea acestui câmp să poată fi mărită și să nu depășească cifra 5.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *char*, numărul de rînduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.

#### **Varianta 12**

a) Să se creeze clasa *Set* – mulțimea numerelor întregi, utilizând memoria dinamică. Să se definească constructorii: implicit și de copiere. Să se definească funcțiile: de adăugare a unui element nou în mulțime; de determinare a apartenenței elementului la mulțime; de adunare, scădere și intersecție a două mulțimi.

b) Să se creeze clasa *Matrix* – matrice. Clasa conține pointer spre *Complex*, numărul de rînduri și de coloane și o variabilă – codul erorii. Să se definească constructorul fără parametri (constructorul implicit), constructorul cu un parametru – matrice pătrată și constructorul cu doi parametri – matrice dreptunghiulară ș. a. Să se definească funcțiile membru de acces: returnarea și setarea valorii elementului (i,j). Să se definească funcțiile de adunare și scădere a două matrice; înmulțirea unei matrice cu alta; înmulțirea unei matrice cu un număr. Să se testeze funcționarea clasei. În caz de insuficiență de memorie, necorespondență a dimensiunilor matricelor, depășire a limitei memoriei utilizate să se stabilească codul erorii.