§2. Clase şi obiecte în C++.

În forma cea mai generală o clasă poate fi descrisă astfel:

```
class [nume_clasa]
{
//membrii clasei
};
```

unde membrii clasei sunt de două tipuri:

- date membre;
- funcții membre.

Cu ajutorul datelor membre pot fi descrise atributele care caracterizează proprietățile obiectelor reale, modelate în cadrul programelor create.

Funcțiile membre descriu funcționalitățile claselor de obiecte din domeniul de problemă.

Forma generală de descriere a unei clase este următoarea:

```
class [nume_clasa] //identificator
{
[specificator_de_acces_a1:]
lista_membri_1;
specificator_de_acces_a2:
lista_membri_2;
- - - - - - - - - - -
specificator_de_acces_aN:
lista_membri_N;
}  [lista_obiecte] ;
[descrierea_funcţiilor_membre_si_prietene]
```

unde `nume_clasa` – un identificator, numele clasei,

`lista_obiecte` – variabile de tip clasă, sau obiecte ale clasei descrise.

În această formă de declarare a unei clase, componentele incluse în paranteze pătrate sunt opționale (pot fi omise).

Componentele din descriere `specificator_de_acces_a1`, `specificator_de_acces_a2`, ..., `specificator_de_acces_aN`, se numesc specificatori de acces şi indică gradul de acces la membrii clasei. Fiecare membru al clasei se află sub acțiunea unui specificator de acces. Specificatorii de acces sunt descrişi prin cuvintele cheie `private`, `protected` şi `public` (rom.: acces privat, protejat, public), urmate de semnul „:”

Următorul tabel descrie gradul de acces permis de fiecare specificator:

	Funcţii membre ale clasei date	Funcţii prietene	Funcţii membre ale claselor derivate	Funcţii externe
private	+	+	–	–
protected	+	+	+	–
public	+	+	+	+

După cum se vede, declarația începe cu cuvântul-cheie **class**, după care urmează denumirea clasei.

Denumirea clasei este orice identificator creat de către programator conform sintaxei limbajului C++ şi care este liber în momentul declarării.

Apoi, urmează corpul clasei luat în acolade { şi }.

Corpul poate fi şi vid.

Corpul clasei poate conține:

- declarări de câmpuri (date) membre,
- declarări de prototipuri ale funcțiilor membre şi ale funcțiilor prietene,
- definiri de funcții membre şi de funcții prietene,
- declarări de clase prietene.

Declararea câmpurilor şi a funcţiilor membre se repartizează pe secţiuni în dependenţă de regimul dorit de accesare a lor.

Acest regim este stabilit prin utilizarea a trei specificatori de acces: **public**, **protected**, **private**. Ordinea secţiunilor, precum şi numărul de repetări sunt arbitrare.

Dacă la începutul corpului de declarare a clasei, pentru un set de câmpuri şi funcţii membre specificatorul de acces nu este indicat, implicit, pentru câmpurile şi funcţiile membre din această secţiune (prima secţiune) va fi stabilit regimul de accesare **private** (privat) – cel mai dur regim de accesare din cele trei. Câmpurile şi funcţiile membre private sunt accesibile numai din interiorul acestei clase, adică acces la ele au numai funcţiile membre ale sale, sau funcţiile de tip prieten (**friend**) al acestei clase.

Regimul de accesare **protected** (protejat) este puțin mai liber în comparație cu regimul **private**.

El stabileşte că câmpurile şi funcțiile membre ale clasei definite vor fi accesibile şi din interiorul claselor derivate ale clasei definite.

Specificatorul de acces **public** este cel mai liber din cei trei.

El admite accesarea câmpurilor şi a funcțiilor membre ale acestei secțiuni din orice loc al programului unde va fi vizibil obiectul concret al clasei definite.

Aşa o accesare este asigurată prin intermediul numelui acestui obiect.

ă

În C++ declararea claselor este similară cu declararea structurilor. În cazul structurilor se foloseşte cuvântul-cheie **struct** în loc de **class**.

O deosebire constă în aceea că în cazul structurilor, implicit, va fi stabilit regimul de accesare **public** şi nu **private**, cum a fost descris mai sus pentru clase.

Imediat după descrierea corpului clasei (acolada de închidere) putem crea un set de obiecte ale acestei clase.

Însă, acest lucru nu este obligatoriu.

Obiectele pot fi create şi mai târziu în program.

După caracterul '**;**' urmează definițiile (realizările) funcțiilor membre, dacă prototipurile lor au fost declarate în corpul clasei.

Exemplu 1. Declarăm versiunea simplificată a clasei pentru reprezentarea fracțiilor fără semn, cu numărător şi numitor – numere naturale (numere raționale pozitive). Vom numi această clasă **fractie**:

```
#include <conio.h>
#include <iostream.h>
class fractie
{
   protected:
      unsigned int numarat;
      unsigned int numit;
};
void main()
{
   fractie fr1; // creăm un obiect numit fr1,
                // reprezentantul clasei fractie

   fractie fr2; // mai creăm un obiect numit fr2,
                // reprezentantul aceleiaşi clase

   fr1 = fr2;   // atribuim obiectului fr1 obiectul fr2
   char c; cin >> c;  // se aşteaptă apăsarea oricărei
                      // taste
}
```

De fapt, conform sintaxei, o versiune minimală a clasei fractie este:

```
class fractie
{
};
```

Însă, fiind perfect validă din punctul de vedere al sintaxei, ea nu este interesantă din punctul de vedere al semanticii, de aceea nici nu va fi examinată.

Clasa `fractie` conține două câmpuri: `numarat` şi `numit` pentru păstrarea, respectiv, a numărătorului şi a numitorului ce alcătuiesc o fracție rațională fără semn. Ambele câmpuri sunt de tipul **`unsigned int`** şi au unul şi acelaşi tip de acces **`protected`**. Dacă înlăturăm linia:

`protected:`

din programul de mai sus, atunci, implicit, câmpurile `numarat` şi `numit` vor avea tipul de acces **`private`**.

Însă, în ambele cazuri, cu obiectele create ale clasei `fractie` putem doar să atribuim un obiect altuia, cum este arătat în funcția `main()`. Operatorul de atribuire este implicit realizat cu orice clasă declarată. El presupune copierea datelor (valoarea numărătorului şi numitorului) din a doua fracție în prima.

Nu se ştie ce va fi copiat în `fr1` în exemplul nostru, fiindcă nici în câmpurile obiectului `fr1`, nici în câmpurile obiectului `fr2` nu a fost înscrisă nici o valoare până la aplicarea operației de atribuire. Inițializarea nu a fost prevăzută în versiunea clasei de mai sus. Mai mult ca atât, nici nu există posibilitatea de a înscrie cumva valori în câmpurile obiectelor `fr1` şi `fr2` (din cauza că sunt protejate). Unicul lucru care poate fi afirmat este că după atribuire, conținuturile câmpurilor obiectelor `fr1` şi `fr2` vor fi identice.

Din cauza că câmpurile clasei sunt protejate, orice încercare de accesare, cum ar fi, de exemplu:

```
fr1.numarat = 2;
fr2.numit = 3;
```

vor fi respinse de către compilator, fiind interzise (se va semnala o eroare).

În versiunea propusă a clasei `fractie` nu putem iniţializa câmpurile obiectelor create, nu putem afişa obiectele, nu putem efectua calcule asupra lor etc. Aceste acţiuni se pot realiza cu ajutorul funcţiilor membre (metodelor) ale clasei.

Exemplu 2. De alcătuit un program în care este descrisă clasa **carte.** În baza acestei clasei sunt create obiecte, care sunt apoi utilizate, apelând la metodele clasei **initializare** şi **afisare**.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
class carte
{    char* autor;
     char* denumire;
     int an_ed;
  public:
     void initializare (char* a,
                        char* d, int ae);
     void afisare();
     void nimicire();
};
```

```
void carte::initializare(char* a,char* d,
                                int ae)
{
     autor=(char*)malloc(srtlen(a)+1);
     strcpy(autor,a);
     denumire=(char*)malloc(strlen(d)+1);
     strcpy(denumire,d);
     an_ed=ae;
}
```

```
void carte::afisare()
{
    printf(”Autorul cartii:
           %s\n”,autor);
    printf(”Denumirea cartii:
           %s\n”,denumire);
    printf(”Anul editarii:
           %d\n”,an_ed);
}
```

```
void carte::nimicire()
{
    free (autor);
    free (denumire);
}
```

```
void main ()
{
     class carte em, an;
     em.initializare(”M.Eminescu”,
                ”Opere complete”,2002);
     an.initializare(”A.Demidovici”,
          ”Analiza matematica”,1960);
     em.afisare();
     an.afisare();
     em.nimicire();
     an.numicire();
}
```

În rezultatul îndeplinirii acestui program se va afişa pe ecran:

```
Autorul cartii: M.Eminescu
Denumirea cartii: Opere complete
Anul editarii: 2002
Autorul cartii: A.Demidovici
Denumirea cartii: Analiza matematica
Anul editarii: 1960
```

(~§2)