6-MA’RUZA.
OBYEKTLARNING O’ZARO ALOQASI. BOG’LANISH TURLARI

1. Obyektlar orasidagi bog‘lanish

Hayotimiz obyektlar orasidagi takrorlanadigan shablonlar, munosabatlar va iyerarxiyalarga to‘la. Ularni o‘rganib, ularning hayotda qanday ishlashini va bir -biri bilan o‘zaro munosabatini chuqurroq tushunamiz.

Masalan, ko‘chada ketayapmiz va yashil uzun bo‘yga ega yorqin sariq rangni ko‘ramiz. Tushunamizki, yorqin sariq rang - bu gul, yashil esa - poyasi. Agar ilgari bunday o‘simlikni ko‘rmagan bo‘lsak ham, hali ham poyadagi narsalar quyosh nurlari bilan o‘zaro ta'sir qiladigan barglar ekanligini tushunamiz va gul o‘simlikning ko‘payishiga va omon qolishiga yordam beradi. Bilamizki, agar o‘simlik payxon qilinsa, gul ham o‘ladi.

Qanday qilib buni bilamiz va bu obyekt bilan hech qachon uchrashmasdan o‘simlik sifatida hukm qila olamiz? Gap shundaki, o‘simliklar haqida bilimga egamiz va ko‘cha-ko‘yda uchrashgan narsa aynan o‘simliklarga tegishli ekanligini tushunamiz.

Bilamizki, koʻpchilik oʻsimliklarning barglari bor, ba'zilarida esa gullar bor. Barglar quyosh nuri bilan oʻzaro ta'sirlashishini ham bilamiz (hatto bu oʻzaro ta'sir jarayonini toʻliq tushunmasak ham) va gulning mavjudligi toʻgʻridan-toʻgʻri oʻsimlikka bogʻliq va bularning barchasini bilganimizdan va bu oʻsimliklarga taalluqli boʻlgani uchun, oʻsimliklar haqidagi bilimimizga mos keladigan koʻchadagi obyekt haqida shunday xulosalar chiqarishimiz mumkin.

Xuddi shunday, dasturlash ham takrorlanuvchi shablonlar, munosabatlar va iyerarxiyalarga toʻla. Xususan, dasturlashdagi obyektlar haqida gap ketganda, bizni real hayotdagi obyektlarga nisbatan yoʻnaltiradigan bir xil shablonlar oʻzimiz yaratgan dasturlash obyektlariga taalluqlidir. Ushbu munosabatlarni, takrorlanuvchi shablonlarni va iyerarxiyalarni batafsil oʻrganib chiqib, boshqa dasturlarda qayta ishlatish kodini qanday yaxshilashni va osonlik bilan uzaytiriladigan sinflarni qanday yozishni tushunishimiz mumkin.

Dasturlashda obyektlar o‘rtasidagi munosabatlar. Haqiqiy hayotda ikkita obyekt bo‘lishi mumkin bo‘lgan har xil turdagi munosabatlar mavjud va ularni tasvirlash uchun ma'lum so‘zlardan (munosabatlar turlaridan) foydalanamiz, masalan:

-Kvadrat - bu geometrik shakl.

-Mashinaning ruli bor.

-Dasturchi klaviaturadan "foydalanadi".

-Gul o‘simlikka "bog‘liq".

-Talaba guruhning "a'zosi" dir.

-Miyamiz o‘zimizning "qismimiz" sifatida mavjud.

Bu turdagi munosabatlarning hammasi C++ tilida analogga ega.

Quyida biror narsaning a'zosi va qismining borligi, ishlatilishi, bog‘liqligi, nyuanslarini ko‘rib chiqamiz va ular C++ darslari kontekstida qanday foydali bo‘lishi mumkinligini ko‘rsatamiz.

Obyektlar kompozitsiyasi. Haqiqiy hayotda murakkab obyektlar koʻpincha kichikroq, sodda narsalardan iborat. Masalan, mashina metall ramka, dvigatel, toʻrt gʻildirak, rul va boshqa koʻplab qismlardan iborat. Shaxsiy kompyuter markaziy protsessor, xotira va boshqalardan iborat. Oddiy narsalardan murakkab obyektlarni qurish jarayoni *obyektlar kompozitsiyasi* deb ataladi.

Obyektlar kompozitsiyasi turlari. Kompozitsiyada "egalik" munosabatlar turi ikki obyekt oʻrtasida ifodalanadi. Mashinada uzatish qutisi mavjud. Sizning kompyuteringiz markaziy protsessor birligiga "ega". Sizda yurak "bor". Ba'zida murakkab obyekt butunlik (yoki "ajdod") deb nomlanadi. Oddiy obyektni odatda *qism* deb atashadi (yoki "avlod", "komponent").

Ilgari biz strukturalar va sinflar har xil turdagi ma'lumotlarga ega boʻlishi mumkin (masalan, fundamental yoki umuman boshqa sinflar) deb hisoblaganmiz. A'zolar bilan sinflar yaratganimizda, asosan oddiy qismlardan murakkab obyektni yaratamiz, bu obyektning kompozitsiyasi hisoblanadi.

Shu sababli, strukturalar va sinflar tarkibli (kompozitsion) ma'lumotlar turlari deb ham ataladi.

Obyektlar kompozitsiyasi C++ tili kontekstida foydalidir, chunki u sodda va boshqariladigan qismlarni birlashtirib murakkab sinflar yaratishga imkon beradi. Bu murakkablikni kamaytiradi va kodni tezroq va kam xatolar bilan yozishga imkon beradi, chunki biz allaqachon yozilgan, sinovdan o‘tgan va ishlayotgan kodni qayta ishlatishimiz mumkin.

Obyektlar kompozitsiyasining ikkita asosiy kichik turi mavjud: kompozitsiya va agregatsiya. Quyida ularni ko‘rib chiqamiz.

Terminologiya bo‘yicha eslatma: "Kompozitsiya" atamasi ko‘pincha kompozitsiyaning pastki turini emas, balki butun tarkibini va yig‘ilishini bildirish uchun ishlatiladi. Bu darsda biz "obyekt kompozitsiyasi" atamasini butun (kompozitsiya va agregatsiya) ma'nosida, "kompozitsiya" atamasini esa kompozitsiyaning kichik turi haqida gapirganda ishlatamiz.

2. Obyektlar orasidagi bog’lanish turlari

Kompozitsiya. Kompozitsiyani amalga oshirish uchun obyekt va qism quyidagi munosabatlarga ega bo‘lishi kerak:

-Qism (a'zo) - bu obyekt (sinf) ning bir qismi.

-Qism (a'zo) bir vaqtning o‘zida faqat bitta obyektga (sinfga) tegishli bo‘lishi mumkin.

-Qism (a'zo) mavjud, obyekt (sinf) tomonidan boshqariladi.

-Qism (a'zo) obyekt (sinf) borligidan bexabar.

Inson tanasi va yuragi o‘rtasidagi munosabatlar hayotdagi kompozitsiyaning yaxshi namunasidir. Keling, buni batafsil ko‘rib chiqaylik.

Kompozitsiyadagi munosabatlar-bu butun munosabatlar. Masalan, yurak inson tanasining bir qismidir. Kompozitsiyaning bir qismi bir vaqtning o‘zida faqat bitta obyektning bir qismi bo‘lishi mumkin. Bir odam tanasining bir qismi bo‘lgan yurak bir vaqtning o‘zida boshqa odam tanasining bir qismi bo‘la olmaydi.

Kompozitsiya tarkibidagi munosabatlarda obyekt qismlarning mavjudligi uchun javobgardir. Ko‘pincha bu shuni anglatadiki,

qism obyekt yaratilganda va u yo‘q qilinganida yo‘q qilinadi. Ammo keng ma'noda, bu shuni anglatadiki, obyekt qismning ishlash muddatini shunday boshqaradiki, obyektni ishlatayotgan foydalanuvchi unda ishtirok etishi shart emas. Masalan, tana yaratilganda, yurak ham yaratiladi. Qachonki, odamning hayot o‘tsa, uning yuragi ham ishdan chiqadi.

Xullas, qism butun mavjudligini bilmaydi. Sizning yuragingiz katta tashkilotning bir qismi ekanligini bilmay turib, kun bo‘yi ishlaydi. Bu bir tomonlama munosabat deb ataladi, chunki tana yurak haqida biladi, lekin yurak tana haqida bilmaydi.

E'tibor bering, kompozitsiyada qismlarning ko‘chirilishi haqida hech narsa aytilmagan. Yurak bir odamning tanasidan boshqa odamning tanasiga ko‘chirilishi mumkin. Ammo, transplantatsiyadan keyin ham, u kompozitsion talablarga javob beradi (yurak boshqa odamga tegishli va faqat yurak boshqa transplantatsiya qilinmaguncha, boshqa odamning bir qismi bo‘lishi mumkin va boshqa hech kimniki bo‘lmasligi mumkin).

-listingda Drob sinfimiz kompozitsiyaning namunasi keltirilgan:

```
#include <iostream>
using namespace std;
class Drob
{
private:
 int m_numerator;
 int m_denominator;

public:
 Drob(int numerator=0, int denominator=1)

 {
      m_numerator=numerator;
      m_denominator=denominator;
 }
};

int main()
{
```

```
    Drob A;
    return 0;
}
```

Bu sinf ikkita a'zoga ega: m_numerator va m_denominator. Sur'at va maxraj Drobning bir qismidir, ular shu sinfda. Ular bir vaqtning o‘zida boshqa sinfga tegishli bo‘la olmaydi. m_numerator va m_denominator ular Drobning bir qismi ekanligini bilishmaydi, ular faqat butun sonlarni saqlaydi. Drob sinfining obyektini yaratishda m_numerator ham, m_denominator ham yaratiladi. Drob sinfining obyekti o‘chirilgan bo‘lganda, bu a'zolar ham yo‘q qilinadi.

Obyektlar tarkibidagi munosabatlar turi *"bor"* (tanada yurak "bor", Drob sinfida m_denominator "bor") bo‘lgani uchun, biz aytishimiz mumkinki, kompozitsiya *"biror narsaning bir qismi"* munosabatlariga ham ega (yurak - tananing "qismi", m_numerator - Drobning "qismi"). Tarkibi ko‘pincha bir obyekt fizik jihatdan boshqa obyekt ichida bo‘lgan fizik munosabatlarni modellashtirish uchun ishlatiladi.

Kompozitsiyadagi qismlar singular (bir xil) yoki multiplikativ (bir nechta shunday qismlar bo‘lishi mumkin). Masalan, inson tanasida faqat bitta yurak bor (yurak yagona), lekin 10 ta barmoq bor (barmoqlar multiplikativ va ularni massiv sifatida ishlatish mumkin).

Kompozitsiyalarni realizatsiya qilish. Kompozitsiyalar C++ da amalga oshiriladigan munosabatlarning eng oddiy turlaridan biridir. Bu muntazam a'zolari bo‘lgan muntazam struktura yoki sinflar. A'zolar to‘g‘ridan-to‘g‘ri strukturalar/sinflarning bir qismi sifatida mavjud bo‘lganligi sababli, ularning hayoti to‘g‘ridan-to‘g‘ri ushbu strukturalar/sinflar obyektlarining ishlash muddatiga bog‘liq.

Kompozitsiyaning o‘zgarishi (variatsiyasi). Garchi ko‘pchilik kompozitsiyalarda qismlarni yaratish/o‘chirish to‘g‘ridan-to‘g‘ri kompozitsiyani yaratishda/o‘chirishda ro‘y bersa-da, qoidalar biroz o‘zgartirilgan kompozitsiyaning o‘zgarishlari mavjud, masalan:

-Kompozitsiya uning ba'zi qismlarini kerak bo‘lguncha qoldirishi mumkin. Masalan, string sinfi foydalanuvchi saqlay oladigan

ma'lumotlarni bermaguncha dinamik simvollar massivini yaratmasligi mumkin.

-Kompozitsiya o‘zi yaratganidan ko‘ra, kirish sifatida berilgan qismdan foydalanishni afzal ko‘rishi mumkin.

-Kompozitsiya uning qismlarini yo‘q qilishni boshqa obyektga topshirishi mumkin.

Asosiy nuqta shundaki, kompozitsiya foydalanuvchining aralashuvisiz o‘z qismlarini o‘zi boshqarishi kerak.

Agregatsiya. Kompozitsiya haqida gap ketganda obyektlar kompozitsiyasi – bu oddiy obyektlardan murakkab obyektlarni yaratish jarayoni ekanligini aytdik. Shuningdek, biz obyektlar kompozitsiyasining pastki turi – kompozitsiya haqida gaplashdik. Tarkib ichidagi munosabatlarda butun (sinf) qismlarning (a'zolar) mavjudligi uchun javobgardir.

Quyida obyektlar tarkibining ikkinchi kichik turini - *agregatsiya*ni ko‘rib chiqamiz.

Agregatsiyani amalga oshirish uchun butun va uning qismlari quyidagi munosabatlarga mos kelishi kerak:

Bir qism (a'zo) - bu butun (sinf) ning bir qismi.

Bir qism (a'zo) bir vaqtning o‘zida bir nechta butun (sinf) ga tegishli bo‘lishi mumkin.

Bo‘lim (a'zo) mavjud, butun (sinf) tomonidan nazorat qilinmaydi.

Qism (a'zo) butun (sinf) ning mavjudligidan xabardor emas.

Kompozitsiyaning pastki turida bo‘lgani kabi, agregatsiyadagi munosabatlar ham butun-butun munosabatlardir va bir tomonlama bo‘ladi. Biroq, kompozitsiyadan farqli o‘laroq, qismlar bir vaqtning o‘zida bir nechta bir butunga tegishli bo‘lishi mumkin va butun qismlarning mavjudligi va yashash vaqtini boshqarmaydi. Agregatsiyani yaratishda / yo‘q qilishda, uning qismlari yaratilishi / yo‘q qilinishi uchun butun javobgar emas.

Masalan, odam va uning uy manzili o‘rtasidagi munosabatni ko‘rib chiqaylik. Har bir insonning o‘z manzili bor. Biroq, bu manzil bir vaqtning o‘zida bir nechta odamga tegishli bo‘lishi mumkin, masalan, siz va sizning xonadoshingiz yoki siz bilan yashaydigan qarindoshlaringiz. Bundan tashqari, bu manzilni shaxs boshqarmaydi - bu manzil odam ko‘chib kelishidan oldin bo‘lgan va u ko‘chib ketganidan keyin ham mavjud bo‘ladi. Qolaversa,

odam qaysi manzilda yashayotganini biladi, lekin manzil, o‘z navbatida, u qanday odam ekanligini va umuman, ularning qanchasi borligini bilmaydi. Bunday munosabatlar *agregatsiyadir*.

Shu bilan bir qatorda, mashina va dvigatelni ko‘rib chiqaylik. Dvigatel avtomobilning bir qismidir. Dvigatel mashinaga tegishli bo‘lsa-da, u boshqa narsalarga ham tegishli bo‘lishi mumkin, masalan, mashinaning egasi. Dvigatelni yaratish yoki yo‘q qilish uchun mashina javobgar emas. Va shu bilan birga, mashina dvigateli borligini biladi (axir, u harakat qiladi), lekin dvigatelning o‘zi uning mashinaning bir qismi ekanligini bilmaydi.

Fizik obyektlarni modellashtirish haqida gap ketganda, "yo‘qotish" atamasini ishlatish biroz noaniq bo‘lishi mumkin. Savol tug‘iladi: "Agar meteorit osmondan tushib mashinani ezib tashlasa, unda mashinaning barcha qismlari ham vayron bo‘ladi deb taxmin qilish mumkinmi?" Ha albatta. Lekin bu mashinaning emas, meteoritning aybidir. Muhim nuqta shundaki, mashina uning qismlarini yo‘q qilish uchun javobgar emas (lekin bunga hissa qo‘shadigan tashqi kuch ham bor).

Aytishimiz mumkinki, agregatdagi munosabatlar turi ham *"bor"* (bo‘limda xodimlar "bor", mashinada dvigatel "bor").

Kompozitsiya singari, agregat qismlari ham yakka (singular) yoki ko‘paytirilishi(multiplikativ) bo‘lishi mumkin.

Agregatsiyani amalga oshirish. Agregatsiya kompozitsiyaga o‘xshash bo‘lgani uchun hamda ikkalasi ham butun-butun munosabatlardan iborat bo‘lgani uchun, ular deyarli bir xil tarzda amalga oshiriladi va ularning orasidagi farq asosan semantikdir. Kompozitsiyada biz oddiy a'zo o‘zgaruvchilari (yoki sinfda xotirani dinamik ravishda ajratish/bo‘shatish ko‘rsatgichlari) yordamida qismlar qo‘shamiz.

Agregatsiyada a'zo o‘zgaruvchilar yordamida qismlar qo‘shamiz. Biroq, bu a'zo o‘zgaruvchilari odatda sinfdan tashqarida yaratilgan obyektlarni ko‘rsatadigan havolalar yoki ko‘rsatkichlardir. Shunday qilib, agregatsiya parametr sifatida ko‘rsatadigan qismlarni konstruktorga oladi yoki agar parametrlar bo‘lmasa, qismlar keyinchalik qo‘shimcha funksiyalar yoki qo‘shimcha yuklangan operatorlar orqali qo‘shiladi.

Bu qismlar sinf doirasidan tashqarida bo‘lgani uchun, sinf yo‘q qilinganda, mos yozuvlar yoki ko‘rsatkichlar ko‘rinishidagi a'zolar o‘zgaruvchilari ham yo‘q qilinadi (lekin ular ko‘rsatadigan qiymatlar o‘chirilmaydi). Shunday qilib, qismlarning o‘zi mavjud bo‘lishda davom etmoqda.

Xodim va bo‘limning misolini batafsil ko‘rib chiqaylik. Ishni osonlashtirish uchun bo‘limda faqat bitta ishchi bor va u qaysi bo‘lim ishchisi ekanligini bilmaydi:

```
#include <iostream>
#include <string>
using namespace std;
class Worker
{
private:
    string m_name;

public:
    Worker(string name)
    {
```

```
        m_name = name;
    }

    string getName()
    {
        return m_name;
    }
};

class Department
{
private:
    Worker *m_worker; // osonlashtirish uchun bu bo‘limda faqat bitta xodim ishlaydi, lekin bir nechta bo‘lishi mumkin

public:
    Department(Worker *worker = nullptr)
    {
        m_worker = worker;
```

```
    }
};

int main()
{
    //Department sinfining ko‘rinish sohasidan tashqarida Ishchi yarating
    Worker *worker = new Worker("Anvar"); //Ishchi yaratish
    {
        // Department yarating va ishchini bo‘limga parametrli konstruktor orqali jo‘nating
        Department department(worker);

    } // department ko‘rinish sohasidan chiqadi va bu yerda yo‘q qilinadi

    // worker mavjud bo‘lishda davom etmoqda

    cout << worker->getName() << " hali ham mavjud!";
```

```
    delete worker;

    return 0;
}
```

Assotsiatsiya. Yuqorida obyektlar kompozitsiyasining ikkita kichik turini ko‘rib chiqdik: kompozitsiya va agregatsiya. Obyektlar tarkibi murakkab obyekt (butun) bir nechta oddiy obyektlardan (qismlardan) iborat bo‘lgan munosabatlarni modellashtirish uchun ishlatiladi.

Endi o‘zaro bog‘liq bo‘lmagan ikkita obyekt - assotsiatsiya o‘rtasidagi munosabatlarning navbatdagi turini ko‘rib chiqamiz. Obyektlar kompozitsiyasidan farqli o‘laroq, assotsiatsiyada butun-butun munosabatlar mavjud emas.

Assotsiatsiyada ikkita bog‘liq bo‘lmagan obyekt quyidagi munosabatlarga mos kelishi kerak:

-Birinchi obyekt (a'zo) ikkinchi obyekt (sinf) bilan bog‘liq emas.

-Birinchi obyekt (a'zo) bir vaqtning o‘zida bir nechta obyektlarga (sinflarga) tegishli bo‘lishi mumkin.

-Birinchi obyekt (a'zo) mavjud, ikkinchi obyekt (sinf) tomonidan boshqarilmaydi.

Birinchi obyekt (a'zo) ikkinchi obyekt (sinf) ning mavjudligi haqida bilishi mumkin yoki bilmasligi mumkin.

Kompozitsiya yoki agregatsiyadan farqli o‘laroq, qism bir butunning qismi bo‘lsa, assotsiatsiyadagi narsalar bir-biri bilan bog‘liq emas. Agregatsiya kabi birinchi obyekt bir vaqtning o‘zida bir nechta obyektlarga tegishli bo‘lishi mumkin va ular tomonidan boshqarilmaydi. Biroq, munosabatlar bir tomonlama bo‘ladigan agregatsiyadan farqli o‘laroq, munosabatlar bir tomonlama yoki ikki tomonlama bo‘lishi mumkin (har ikkala obyekt ham bir-birining mavjudligidan xabardor bo‘lganda).

Shifokorlar va bemorlar o‘rtasidagi munosabatlar assotsiatsiyaning ajoyib namunasidir. Shifokor bemor bilan bog‘liq, ammo bu munosabatni butun-butun munosabatlar deb atash mumkin emas. Shifokor kuniga o‘nlab bemorlarni, bemor esa bir nechta shifokorlarni ko‘rishi mumkin.

Aytishimiz mumkinki, assotsiatsiyadagi munosabatlar turi *"ishlatadi"*. Doktor bemorni daromad olish uchun "ishlatadi". Bemor kasallikdan davolanish yoki farovonligini yaxshilash uchun shifokorni "ishlatadi".

Assotsiatsiyalarni amalga oshirish. Assotsiatsiya turli yo‘llar bilan amalga oshiriladi. Biroq, ko‘pincha ular ko‘rsatgichlar orqali amalga oshiriladi, bu yerda sinflar bir-birining obyektlarini ko‘rsatadi.

Kompozitsiya, agregatsiya va assotsiatsiya o‘rtasidagi farqni tezda tushunishga yordam beradigan jadval:

Xossa	**Kompozitsiya**	**Agregatsiya**	**Assotsiatsiya**
Munosabat	Butun-qism	Butun-qism	Obyektlar bir-biri bilan bog‘liq emas
A'zolar bir vaqtning o‘zida bir nechta sinflarga tegishli bo‘lishi mumkin	Yo‘q	Ha	Ha
A'zolar mavjudligi sinflar tomonidan boshqariladi	Ha	Yo‘q	Yo‘q
Munosabat turi	Bir tomonlama	Bir tomonlama	Bir tomonlama yoki ikki tomonlama

Munosabat tipi	"Biror narsaning bir qismi"	"Unda bor"	"Foydalanadi"

Bog‘liqlik. C++ tilida munosabatlarning 3 turini ko‘rib chiqdik: kompozitsiya, agregatsiya va assotsiatsiya. O‘zaro munosabatlarning eng oddiy turi - bu bog’liqlik.

Kundalik hayotda biz "bog’liqlik" atamasidan foydalanib, bitta obyekt ma'lum bir vazifa uchun ikkinchi obyektga bog‘liqligini bildiramiz. Misol uchun, agar siz oyog‘ingizni sindirib qo‘ysangiz, siz tayoqchalarga tayanasiz (lekin aksincha emas). Gullaydigan o‘simliklar ko‘paytirish uchun ularni changlatishi asalarilarga bog‘liq (lekin aksincha emas).

Bog‘liqlik bitta obyekt boshqa vazifani bajarish uchun ma'lum bir vazifani bajarish uchun kirganda paydo bo‘ladi. Bu munosabatlar assotsiatsiyadagi munosabatlarga qaraganda kuchsizroq, lekin shunga qaramasdan, unga bog‘liq bo‘lgan obyektga uning funksionalligini ta'minlaydigan har qanday o‘zgarish qaram obyektning ishdan chiqishiga olib kelishi mumkin. Bog‘liqlik har doim bir tomonlama.