10- Mavzu: Obyektga yo’naltirilgan dasturlash tillari

Obyektga yo‘naltirilgan dasturlash

Obyektga mo‘ljallangan yondoshuv dasturiy tizimlarni dasturlash tiliga bog‘liq bo‘lmagan holda yaratishda modellardan sistematik foydalanishga asoslangan. Har bir model uning o’zi aks ettirayotgan predmetning hamma xususiyatlarini ifodalay olmaydi, u faqat ba’zi juda muhim belgilarini ifodalaydi. Demak model o’zi aks ettirayotgan predmetga nisbatan ancha sodda bo’ladi. Bizga shu narsa muhimki model endi formal konstruktsiya hisoblanadi: modellarning formalligi esa ular orasidagi formal bog’lanishlarni aniqlashni va ular orasida formal operatsiyalar bajarishni ta’minlaydi. Bu ish modellarni ishlab chiqishni va o’rganishni hamda kompyuterda realizatsiya qilishni osonlashtiradi.
Xususan esa, modellarning formal xarakteri yaratilayotgan dasturning formal modelini olishni ta’minlaydi. Shunday qilib, obyektga mo’ljallangan yondoshuv quyidagi murakkab muammolarni hal qilishda ishlatiladi.
Dasturiy ta’minotning murakkabligini pasaytiradi;
Dasturiy ta’minotning ishonchliligini oshiradi;

Dasturiy ta’minotning a’lohida komponentalarni modifikatsiya qilishni osonlashtiradi.

A’lohida komponentalardan qayta foydalanishni ta’minlaydi.

Obyektga mo’ljallangan yondoshuvning sistemali qo’llanilishi yaxshi tuzilmalangan, ishlatishda barqaror bo’lgan, oson modifikatsiya qilinuvchi dasturiy sistemalarni yaratish imkoniyatini beradi. Aynan ana shu imkoniyatlar dasturchilarni obyektga mo’ljallangan yondoshuvdan foydalanishga juda ham qiziqtirmoqda. Obyektga mo’ljallangan yondoshuvli dasturlash hozirgi vaqtda eng tez rivojlanayotgan dastur yozish texnologiyasi hisoblanadi. Obyektga mo’ljallangan yondoshuv ikkita kismga bo’linadi:
1.Obyektga mo’ljallangan dasturlar yaratish;
2.Obyektga mo’ljallangan dasturlash tillari.

Obyektga mo’ljallangan dasturlar yaratish

Obyektga mo’ljallangan dasturlar yaratish, dastur yaratishda obyektga mo’ljallangan modellarni yaratishga asoslangan. Obyektga mo’ljallangan dasturlar yaratish deganda biz: dasturiy sistemalarni yaratishdagi obyektga mo’ljallangan metodologiyani, bu texnologiyani qo’llovchi instrumental vositalarni tushunamiz.

Obyektga mo’ljallangan dasturlar yaratish dasturiy vositalarni yaratishning hayotiy tsiklining birinchi boskichidayok qo’llanilishi mumkin va u dasturlash tillariga bog’lik emas. Yaratish jarayonida obyektlar-bu formal konstruktsiyalar bo’lib (masalan, burchaklari yoydan tashkil topgan to’rtburchaklar) ularni obyektlar aks ettiradi. Obyektga mo’ljallangan dasturlash yaratish demak obyektga mo’ljallangan metodologiyani(texnologiyani) qo’llashga asoslangan.

Obyektga mo’ljallangan dasturlash tillariga oxirgi vaqtlarda juda ommaviylashgan dasturlash tillari kiradi. Bular quyidagilar: C++, Visual C++, Visual Basic.NET, Java va boshqalar. C++ eng ko‘p tarqalgan obyektga mo‘ljallangan dasturlash tillariga kiradi.

Obyektga mo‘ljallangan dasturlashda dastur obyektlarni va ularning xususiyatlarini(atributlarini) va ularni birlashtiruvchi sinflarni tavsiflashga olib kelinadi. Shu jumladan obyektlar ustida operatsiyalar (usullar) aniqlashga olib kelinadi. Atributlar va usullarni tadqiq qilish asosida bazaviy sinflar va ularning hosilalarini yaratish imkoniyati to‘g‘iladi.

Sinf va obyekt tushunchasi

Sintaksis bo‘yicha, C++ da sinf – bu mavjud bo‘lgan tiplar asosida yangi yaratilgan strukturlangan tip.
Sinf ta’rifi sodda shakli:
<sinf_tipi> <sinf_nomi>{<sinf_komponentlari_ro‘yxati>};
bu yerda:
sinf_tipi –class, struct, union xizmatchi so‘zlaridan biri;

Vorislik, inkapsulyatsiya, polimorfizm.

Obyektga mo‘ljallangan dasturlashning yana bir nazariy jihatdan juda muhim va zarur xususiyatlaridan biri hodisalarni ishlash mexanizmi hisoblanadi, ular yordamida obyektlar atributlari qiymatlari o‘zgartiriladi. Obyektga mo‘ljallangan dasturlashda avval yaratilgan obyektlar bibliotekasi va usullaridan foydalanish hisobiga obyektga yo‘naltirilgan dasturlashda ancha mehnat tejaladi.
Obyektlar, sinflar va usullar polimorfizm bo‘lishlari mumkin, bu esa DV ning qo‘lay foydalanishligi va universalligini ta’minlaydi.
1.Vorislik
2.Inkapsulyatsiya (usullar va xususiyatlarni obyekt ichida saqlash-yashirish).
3.Polimorfizm, berilmalarni ishlash funksiyalarining mavjudligi.
4. Abstraktsiya.Abstraktsiya – bu identifikatorlardan farqli bo‘lgan istalgan dasturlash tili ifodasi hisoblanadi.
Obyektga mo‘ljallangan dasturlashda har bir obyekt printsipial dinamik mohiyatga ega, ya’ni u vaqtga bog‘lik holda va unga nisbatan tashqi faktorlar ta’sirida o‘zgaradi. Boshqacha aytganda obyekt ma’lum bir darajada o‘zini tutishiga ega.
Obyektga mo‘ljallangan dasturlashda abstraktsiya OYD ning modeli hisoblanadi.
Sinf umumiy xususiyatlar va hulk-atvorga ega bo‘lgan obyektlarni birlashtiradi.
Bitta sinfga mansub obyektlar bir xil xususiyatlarga ega bo‘lib, bir xil xatti-xarakat namoyon etadi.

Sinflar shablon

Sinflar shablon (qolip)ga o‘xshaydi: ular obyektlarning ekzemplyarlarini tayyorlash uchun qo‘llanadi. Belgilar - sinfning tashqaridan ko‘rinib turgan xususiyatlari. Obyekt ichki o‘zgaruvchiga bevosita kirishni takdim etganda yoki

usul yordamida qiymatni kaytargandagina, o‘z belgilarini namoyon kilishi mumkin.
Hulk-atvor - xabarga yoki holatning o‘zgarishiga javoban obyekt tomonidan bajariladigan xatti-xarakatlar. U obyekt nima qilayotganini bildiradi.
Bir obyekt ikkinchi obyekt ustida xatti-xarakatlar bajarib, uning xulk-atvoriga ta’sir ko‘rsatishi mumkin. «Xatti-xarakat» atamasi o‘rniga «usulni chakirish», «funksiyasini chakirish» yoki «xabarni o‘zatish» atamalari ko‘llanadi. Muximi bu atamalarning qaysi biri qullanayotganida emas, albatta, muximi bu xatti-xarakatlar obyekt hulk-atvorini namoyon qilishga da’vat etishidadir. Obyektlar o‘rtasida aloqa obyektga mo‘ljallangan dasturlashning muhim tarkibiy qismidir. Obyektlar o‘zaro aloqasining ikkita asosiy usuli mavjuddir.

Birinchi usul: obyektlar biri ikkinchisidan mustaqil ravishda mavjud bo‘ladi. Agar alohida obyektlarga o‘zaro aloqa kerak bo‘lib qolsa, ular bir-birlariga xabar jo‘natadi. Obyektlar bir-birlari bilan xabarlar yordamida aloqa qiladi. Xabar olgan obyekt ma’lum xatti-xarakatlarni bajaradi. Xabar uzatish bu obyekt xolatini o‘zgartirish maqsadida uslubni chaqirib olish yoki xulk-atvor modellaridan birini ko‘llashning o‘zginasidir.

Ikkinchi usul: obyekt tarkibida boshka obyektlar bo‘lishi mumkin. Xuddi OMDda bo‘lganidek, dastur obyektlardan tashkil topganidek, obyektlar ham, o‘z navbatida, agregattsiya yordamida boshqa obyektlardan jamlanishi mumkin. Ushbu obyektlarning har bittasida uslub va belgilarga ega bo‘lgan interfeys mavjud bo‘ladi.
Xabar - obyektga mo‘ljallangan yondoshuvning muhim tushinchasi. Xabarlar mexanizmi tufayli obyektlar o‘z mustakilligini saqlab qolishi mumkin. Boshqa biron obyektga xabar jo‘natayotgan obyekt uchun xabar olgan obyekt talabdagi xatti-xarakatni qanday bajarishi unchalik muhim emas. Unga xatti-xarakat bajarilganligining o‘zi muhimdir.

Obyektga yo‘naltirilgan dasturlash tillari

C++ dasturlash tili
C++ da quyidagi ma’lumot turlari ishlatiladi: int, float, double, char -belgili tur, string – belgilar qatori.
O‘zgaruvchilarni e’lon qilish quyidagicha amalga oshiriladi:
float x1, x2, x3, y1, y2; int d1, d2;
O‘zguruvchilarni e’lon qilish va initsializatsiya qilishni bitta operator bilan amalga oshirish ham mumkin: int nVariable = 1;
O‘zgarmas deb istalgan doimiy kattalikga aytamiz. O‘zgaruvchilar singari o‘zgarmaslar ham turlarga ega:
const int a=25; const float b=12.27; const char plus=’+’; const string b=”Rezult”;
Istalgan chop kilinuvchi belgilar bilan ishlash uchun char yoki string turidagi o‘zgaruvchilardan foydalanish mumkin. C++ da bitta ifodada har xil turdagi o‘zgaruvchilar ishlatilishi mumkin:

O‘zgaruvchilar ustida operatsiyalar bajarish mumkin: ko‘shish, ko‘paytirish, ayirish, bo‘lish va hokaza

Bu yerda main – dasturning bosh funksiyasining nomi. C++ dasturining bajarilishi hamisha shu funksiyadan boshlanadi. Bu funksiya nomi bor (main), nomdan keyin aylana qavsda funksiya parametrlari keltiriladi. Bu funksiya kaytariluvchi natijaga ega. Bu erda return 0 operatori, funksiya 0 qiymatni qaytarishini bildiradi.

Delphi dasturlash tili

Delphi tili obyektga yo‘naltirilgan dasturlash tilidir. Obyektga yo‘naltirilgan tilga yig‘ilgan imkoniyatlarga dasturlash tilining obyekt modellari deyiladi. Delphi tilida obyekt modellari ishlatilishining amaliy natijasi komponentalarni yaratish va ularni qo‘llab quvvatlashdir.

Obyektga yo‘naltirilgan dasturlash (OYD) – bu dastur ishlab chiqish usuli bo‘lib, uning asosida real dunyo obyekti va uning holatini ifodalovchi ma’lum strukturaga ega obyekt tushunchasi yotadi. Delphi tilida obyekt modelining qo‘llanilish natijasi bu komponentalarni qo‘llash va yaratishdir. Obyektga dasturlash asosi sinf va obyekt tushunchalaridir.

Java dasturlash tili

Java dasturlash tili — eng yaxshi dasturlash tillaridan biri bo‘lib unda korporativ darajadagi mahsulotlarni(dasturlarni) yaratish mumkin.Bu dasturlash tili Oak dasturlash tili asosida paydo bo‘ldi. Oak dasturlash tili 90-yillarning boshida Sun Microsystems tomonidan platformaga (Operatsion tizimga) bog‘liq bo‘lmagan holda ishlovchi yangi avlod aqlli qurilmalarini yaratishni maqsad qilib harakat boshlagan edi. Bunga erishish uchun Sun hodimlari C++ ni ishlatishni rejalashtirdilar, lekin ba’zi sabablarga ko‘ra bu fikridan voz kechishdi. Oak muvofaqiyatsiz chiqdi va 1995-yilda Sun uning nomini Java ga almashtirdi, va uni WWW rivojlanishiga hizmat qilishi uchun ma’lum o‘zgarishlar qilishdi.

Java Obyektga Yo‘naltirilgan Dasturlash (OOP-object oriented programming) tili va u C++ ga ancha o‘xshash. Eng ko‘p yo‘l qo‘yildigan xatolarga sabab bo‘luvchi qismalari olib tashlanib, Java dasturlash tili ancha soddalashtirildi. Java kod yozilgan fayllar (*.java bilan nihoyalanuvchi) kompilatsiyadan keyin bayt kod(bytecode) ga o‘tadi va bu bayt kod interpretator tomonidan o‘qib yurgizdiriladi.

Zamonaviy vizual dasturlash muhitlari

IDE (Integrated development environment) — dasturlash tillari uchun muhit hisoblanadi, ko‘pchilik bu tushunchani aynan shundayligicha biladi, lekin bu ta’rifning ma’nosi nimaligini unchalik tushunmaydi. Bu maqolada aynan shu IDE abreviaturasini tushuntirishga bag‘ishlanadi. Dasturlash muhiti deganda, siz yozayotgan kodlarni aynan qayerga yozish kerakligi tushuniladi. Misol uchun, oddiy «Блокнот» ham IDE vazifasini bajarishi mumkin. IDE sifatida, dasturlar yoki dasturlar yig‘indisi ishlatiladi. Zamonaviy va mashhur dasturlash muhitlariga quyidagilarni misol qilish mumkin .

PHPStorm — asosan PHP dasturchilar uchun;
VisualStudio — .Net dasturchilar uchun;
NetBeans — asosan java, php dasturchilar uchun;
PHPDesigner — asosan web(PHP) dasturchilar uchun;
Agar hali ham tushunarsiz bo‘lsa, boshqa mavzuda misol keltiraman. Siz kompyuter tuzatuvchi ustasiz, siz o‘z ishingizni qilishingiz uchun yaxshi sharoit kerak: elektr toki bilan ta’minlagan xona, kerakli qurilamalarga(tester, payalnik, otvyorka..) ega bo‘lishingiz, har hil turdagi ulanuvchi va ulovchi simlar bo‘lishi, kompyuterning asosiy ehtiyot qismlarining nusxasi va boshqalar. Umuman olib qaraganda, bunday sharoit bo‘lmasa ham usta bo‘laverasiz, lekin biror kompyuterni tuzatish uchun ancha vaqt kerak bo‘lib qoladi(kerakli jihozlarni kimdandur so‘rash kerak bo‘ladi, tok o‘chib qolsa, uni kelishini kutish). IDE ham shunday, qanchalik yaxshi va qulay muhit bo‘lsa, ishingiz ham shuncha tez va sifatli bitadi.
Mukammal dasturlash muhitlarida, dasturchilar uchun hamma sharoitlar yaratilgan bo‘ladi, ya’ni biror loyihani tuzish uchun qo‘shimcha dasturlar kerak bo‘lmasligi lozim, misol uchun quyidagi imkoniyatlar bo‘ladi:
— matn muharriri;
— kompilyator/interpretator;
— loyihaning barcha qismlarini avtomat yig‘uvchi;
— xatolarni aniq ko‘rsatuvchi funksiyasi;
— kod sintaksislarini yozishda yordam beradigan kutubxona;
— kodni ishlatib ko‘rish uchun sharoit(emulyatorlar, brauzerlar);
— terminal(konsol uchun);
— versiyalar bilan ishlovchi modul(github);
— katalog ierarxiyasi;
Bunday dasturlash muhitlari, dasturchilarni biroz dangasa qilib qo‘yadi degan gap rost, lekin tez biror loyihani tuzmoqchi bo‘lsangiz, bularsiz ancha vaqt ketib qoladi.

Dasturlash muhitlari

— PHPStorm;
— VisualStudio;
— RAD Studio;
— KomodoIDE;
— PHPDesigner;
— KomodoEdit
— VS Express;
— NetBeans;
— Aptana Studio;
— Eclipse;
Dasturlash tiliga qarab, kerakli IDE tanlanadi.
Sinflarni ta’riflash. Komponenta funksiyalar. Komponenta ma’lumotlar
Sintaksis bo‘yicha, C++ da sinf – bu mavjud bo‘lgan tiplar asosida yangi yaratilgan strukturlangan tip.

Sinf ta’rifi sodda shakli

Funksiya – bu obyektlar ustida bajariladigan operatsiyalarni aniqlovchi sinf usuli. Ma’lumotlar – bu obyekt strukturasini xosil qiluvchi maydon.
Usullar sinfdan tashqarida aniqlanganda ularning nomlarini kvalifikatsiya qilish (ixtisoslashtirish) kerak. Usulning ko‘rimlilik soxasini aniqlaydigan uning bunday kvalifikatsiya sintaksisi quyidagi ko‘rinishga ega:
<sinf nomi>::<usul nomi>
Sinf ichida aniqlangan usullar ko‘zda tutilgan bo‘yicha joylashtiriluvchi (inline) funksiya hisoblanadi. Sinf tashqarisida aniqlangan usullarni oshkor ravishda joylashtiriluvchi deb ta’riflanishi lozim.
Sinf obyekti (sinf nusxasi) ni ta’riflash uchun quyidagi konstruksiyadan foydalaniladi:
<sinf_nomi> <obyekt_nomi>;
Obyekt orqali maydonlarga va usullarga quyidagicha murojjat qilish mumkin:
< obyekt_nomi >. <maydon_nomi>
< obyekt_nomi >. <usul_nomi>
Murojaat huquqlari
Komponentalarga murojaat huquqi murojaat spetsifikatorlari yordamida boshkariladi: public, private, protected. Umumiy (public) komponentalar dasturni ixtiyoriy qismida murojaat huquqiga ega. Ulardan, ixtiyoriy funksiya ushbu sinf ichida va sinf tashqarida foydalansa ham bo‘ladi. Xususiy (private) komponentalar sinf ichida murojaat huquqiga ega, lekin sinf tashqarisidan esa murojaat qilish mumkin emas. Komponentalardan ushbu ular tavsiflangan sinfdagi funksiya - a’zolari yoki “do‘stona”- funksiyalar orqali foydalanish mumkin.

Ximoyalangan komponentalar

Ximoyalangan komponentalar sinf ichida va xosila sinflarda murojaat huquqiga ega. Agar sinf ta’rifida class so‘zi ishlatilgan bo‘lsa hamma komponentalari xususiy hisoblanadi, agar struct vso‘zi ishlatilgan bo‘lsa hamma komponentalar umumiy hisoblanadi.
Konstruktor va destruktor. Sinf statik komponentlari
Konstruktor - bu sinf obyektlarini avtomatik initsializatsiya qilish uchun ishlatiladigan maxsus komponentali funksiya. Konstruktorlar kurinishi quyidagicha bulishi mumkin:
<Sinf nomi> (<formal parametrlar ruyxati>)
{<konstruktor tanasi>}
Bu komponenta funksiya nomi sinf nomi bilan bir xil bulishi lozim.
Dasturchi tomonidan ko‘rsatilmagan holda ham new operator yordamida sinf obyekti yaratilganda yoki xotirada joylashganda konstruktor avtomatik ravishda chaqiriladi.
Konstruktor obyekt uchun xotirada joy ajratadi va ma’lumotlar – sinf a’zolarini initsializatsiyalaydi.
Konstruktor bir nechta xususiyatlarga ega:
Konstruktorlar uchun kaytariluvchi tiplar, xatto void tipi ham ko‘rsatilmaydi

Konstruktor adresini hisoblash mumkin emas. Konstruktor parametri sifatida uz sinfining nomini ishlatish mumkin emas, lekin bu nomga ko‘rsatkichdan foydalanish mumkin.

Konstruktorlar vorisligi

Konstruktorlar vorislikga ega emas. Konstruktorlar ixtiyoriy sinflar uchun doimo mavjud, lekin agarda u ko‘rsatilgan holda tavsiflanmagan bo‘lsa, u avtomatik ravishda yaratiladi. Ko‘rsatilmagan holda parametrsiz konstruktor va nusxa konstruktori yaratiladi. Agarda konstruktor ochiq holda tavsiflangan bo‘lsa, unda ko‘rsatilmagan holda konstruktor yaratilmaydi. Ko‘rsatilmagan holda umumiy (public) konstruktorlar yaratiladi. Konstruktorni oddiy komponenta funksiya sifatida chakirib bulmaydi. Konstruktorni ikki xil shaklda chakirish mumkin :

Birinchi shakl ishlatilganda xakikiy parametrlar ruyxati bush bulmasligi lozim. Bu shakldan yangi obyekt ta’riflanganda foydalaniladi:
Konstruktorni ikkinchi shaklda chakirish nomsiz obyekt yaratilishiga olib keladi. Bu nomsiz obyektdan ifodalarda foydalanish mumkin.

Sinflarda ko‘rsatkichlar

Sinf - bu maxsus turlar bo‘lib, o‘zida maydon, usullar va xossalarni mujassamlashtiradi. Sinf murakkab struktura bo‘lib, ma’lumotlar ta’riflaridan tashqari, protsedura va funksiyalar ta’riflarini o‘z ichiga oladi. Sodda sinf ta’rifiga misol:

```
TPerson = class
private
fname: string[15];
faddress: string[35];
public
procedure Show;
end;
```

Sinf ma’lumotlari maydonlar, protsedura va funksiyalar usullar deb ataladi.
Keltirilgan misolda TPerson - sinf nomi, fname va faddress – maydonlar nomlari, show - usul nomi.
Maydon - bu sinfga birlashtirilgan ma’lumotlardir. Sinfga qarashli maydonlar oddiy yozuv maydoni kabi bo‘lib, ularning farqi har xil turda bo‘lishidir.

Maydonlarga murojaat qilish

Maydonlarga murojaat qilish sinf xossalari va usullari yordamida amalga oshiriladi. Maydonga murojaat qilish uchun oldin sinf nomi yozilib, keyin ajratuvchi nuqta qo‘yilib maydon nomi yoziladi. Maydon nomi unga mos xossa nomining birinchi harfi “F” bo‘lishi bilan farqlanadi. Delphi da qabul qilingan kelishuv bo‘yicha maydonlar nomlari f (field — maydon so‘zidan) harfidan boshlanishi lozim.

“This” ko‘rsatkichi

Agarda konkret obyektga ishlov berish uchun sinf a’zosi – funksiya chaqirilsa, unda shu funksiyaga obyektga belgilangan ko‘rsatkich avtomatik va ko‘rsatilmagan holda uzatiladi. Bu ko‘rsatqich this ismiga ega va x* this kabi har bir funksiya uchun ko‘rsatilmagan holda belgilanadi.

A’zolarga murojaat etishda this dan foydalanish ortiqcha. Asosan this bevosita ko‘rsatkichlar bilan manipulyatsiya qilish uchun a’zo funksiyalarini yaratilishida foydalaniladi.

Har qaysi obyekt sinf maydonining o‘z nusxasiga ega. Sinf usuli xotirada bitta nusxada bo‘ladi va hamma obyektlar bilan birgalikda ishlatiladi. Shuning uchun usullarni maydonlar bilan ishini tashkil etishda chaqirilayotgan obyekt uchun ta’minlanishiga e’tibor berish kerak. Bu esa yashirin this parametrini funksiyaga o‘zatishni ta’minlaydi. this ko‘rsatkichi noaniq ravishda usulning ichkarisida foydalaniladi. Aniq holatda esa bu ko‘rsatkich usuldagi ko‘rsatkichni hisoblashda (return this;) yoki chaqirilayotgan obyektga (return *this;) murojaatlarda qo‘llaniladi. this ko‘rsatkichini sinf usulning maydonini nomi formal parametrlar nomi bilan mos kelgan hollarda identifikatsiya qilishda qo‘llash mumkin. Identifikatsiyalashning boshqa usuli esa ko‘rinish sohasiga murojaat etishda qo‘llash mumkin:

Self ko‘rsatkichi

Sinfga birlashtirilgan protsedura va funksiyalarga usullar deyiladi. Sinf usullari (sinf ta’rifiga kiritilgan protsedura va funksiyalar) sinf obyektlari ustida amal bajaradi. Usul bajarilishi uchun obyekt nomi va nuktadan sung usul nomi ko‘rsatilishi lozim. Masalan: professor. Show; Sinf usuli ta’riflanganda sinf nomi va usul nomi ko‘rsatiladi. Usul tanasida obyekt maydonlariga murojaat kilinganda obyekt nomi ko‘rsatilmaydi.

Usulga murojaat qilish dasturda uning nomini ko‘rsatish bilan bajariladi. Usullar sinf ichida ta’riflangan protsedura va funksiyalardir. Sinf tarkibiga usullarni chaqirish uchun zarur bo‘lgan ma’lumotlar saqlanuvchi maxsus jadvalga ilova kiradi.Usullar chaqirilganda chaqirgan obyektga ilova uzatiladi. Bu ilovaga usul ichida self so‘zi orqali murojaat qilish mumkin.

Munosabat turlari. Obyektlar sinf a’zolari sifatida

Har bir sinfni ikkita muhim jihati mavjud: u arxitektura birligini moduli hisoblanadi,va u bir necha ma’lumotlar turlarini aniqlagan holda sermazmun tushunchaga ega. Dastur tizimi sinflari hammasi bir-biri bilan o‘zaro aniq bog‘lanishda bo‘ladi.

Obyektli dasturlash tizimida ikkita asosiy tur sinflarni o‘zaro bog‘lanishi aniqlangan.Birinchi bog‘lanish “Mijoz va Yetkazuvchi”, odatda mijoz bog‘lanishi deb ataladi yoki ichma-ich bog‘lanishda bo‘ladi. Ikkinchi bog‘lanish “Ota-onalar va vorislar” bu odatda vorislik deb nomlanadi.

Ta’rif 1. A va V sinflar “mijoz va yetkazuvchi” bog‘lanishida bo‘lsa, agar V sinfning maydoni A sinf obyekti bo‘lsa, A sinf V sinfning yetkazuvchisi deb nomlanadi, V sinfi A sinfning mijozi deb ataladi.

Ta’rif 2. A va V sinflar “Ota-onalar va vorislar” bog‘lanishida deyiladi, agar V sinf e’lon qilishda A sinf ota sinf sifatida ko‘rsatilgan bo‘lsa, V sinf A sinfning vorisi deb ataladi.

Ikkala bog‘lanish – vorislik va ichma-ich joylashganlik tranzitivlik xossasiga ega. Agar V sinf A sinf mijozi, va S sinf V sinfning mijozi, bo‘lsa S sinf A sinfning mijozi bo‘ladi. Agar V sinf A sinf vorisi bo‘lsa, S sinf V sinfning vorisi bщlsa S sinf A sinfning vorisi bo‘ladi.

Yuqoridagi 1 va 2 ta’riflar to‘g‘ridan to‘g‘ri mijoz va yetkazuvchi, vorislik munosabatlarini aniqlaydi (1-bosqich mijoz,1-bosqich yetkazuvchi va hokazo). Rekursiv tarzida shuni aniqlash mumkinki: to‘g‘ridan to‘g‘ri k-chi bosqichdagi mijoz k+1 bosqichdagi mijos bilan bog‘lanishda bo‘ladi. Vorislik bog‘lanishida, tabiiy tildagi tushunchalar qo‘llanadi. To‘g‘ridan to‘g‘ri bo‘lmagan vorislik ajdodlar va avlodlar bog‘lanishi deyiladi. Agar sinfda boshqa sinf obyektlari mavjud bo‘lsa, unda, sinf a’zosi - konstruktor uchun parametri sinf konstruktori ta’rifida (lekin tavsifida emas) ko‘rsatiladi. A’zosi uchun konstruktor uning uchun parametrlar ro‘yxatini belgilovchi konstruktor bajarilgandan so‘ng chaqiriladi.

O‘zaro do‘st funksiyalar va sinflar

Agarda bir nechta sinf funksiyalariga boshqa sinfning xususiy ma’lumotlariga murojaat qilish kerak bo‘lsa, u holda C++ do‘stona sinfning faqatgina belgilangan funksiyalari xususiy elementlarga murojaat etishiga imkoniyat beradi. Masalan, faqatgina change_catalog va get_catalog funksiyalarga book sinfning xususiy elementlariga murojaat kerak.

Quyida ko‘rsatilgandek, book sinfning ichida faqatgina shu funksiyalarda xususiy funksiyalarga murojaat chegarasini qo‘yishi lozim: Agar dastur bir sinfdan boshqasiga murojaat qilsa va sinflar aniqlanish tartibi noto‘g‘ri bo‘lsa sintaksik xatoga duch kelish mumkin. Bizning holda book klassi librarian klassida e’lon qilingan funksiyalar prototiplariga murojat qilmoqda. Shuning uchun librarian klassi aniqlanishi book klassi aniqlanishidan oldin kelishi kerak, biroq librarian klassi book klassiga murojat qilmoqda:

Do‘stona munosabat qoidalari

Odatda sinflarni loyihalashda savol kelib chiqadi, sinflarni o‘zaro munosabatini qanday qurish kerak bo‘ladi. Ikkita oddiy sinflarga misol ko‘ramiz – Square va Rectangle, ular kvadrat va to‘g‘rito‘rtburchaklardir. Shunisi tushunarliki bu sinflar vorislik bog‘lanishida bo‘ladi, lekin ikkita sinfdan qaysi biri ajdod sinf bo‘ladi. Yana ikkita sinfga misol – Car va Person, ya’ni mashina va inson. Bu sinflar bilan Person_of_Car ya’ni mashina egasi sinfi qanday aloqada bo‘lishi mumkin? Bu ikki sinf bilan vorislik bog‘lanishida bo‘lishi mumkinmi? Sinflarni loyihalash bilan bog‘liq bu savollarga javob topish uchun shuni nazarda tutish kerakki, “mijoz-yetkazuvchi” bog‘lanishi “ega” (“has”) bog‘lanishini, vorislik bog‘lanishi esa “bir

xil” (“is a”) bog‘lanishi tushunchalarini ifodalaydi. Square va Rectangle sinflari misoli tushunarli, har bir obyekt kvadrat to‘g‘rito‘rtburchakdir, shuning uchun bu sinflar o‘rtasida vorislik bog‘lanishi ifodalanadi, va Rectangle sinfi ota-onalar sinfini ifodalaydi. Square sinfi uning o‘g‘lidir. Mashina egasi mashinaga ega va insondir. Shuning uchun Person_of_Car sinfi Car sinfning mijozi bo‘lib hisoblanadi va Person sinfning vorisidir.

Sinflarda vorislik

Vorislik g‘oyasi obyektlar xulq-atvorini modifikatsiyalash muammosini hal qiladi hamda OMD ga favqulotda kuch va moslashuvchanlik baxsh etadi. Vorislik, deyarli hech qanday cheklanishlarsiz, siz yoki boshqa biron kimsa tomonidan yaratilgan sinflarni izchil qurish va kengaytirish imkonini beradi. Eng oddiy sinflardan boshlab, murakkablik jihatidan asta-sekin ortib boradigan, ammo sozlanishi ham oson, ichki tuzilishi ham oddiy bo‘lgan hosila sinflarni yaratish mumkin.

Ayniqsa yirik dasturiy loyihalarni ishlab chiqishda vorislik tamoyilini hayotga izchil tatbiq etish pasayib boruvchi tuzilmaviy dasturlash (umumiydan juz’iyga) texnikasi bilan yaxshi moslashadi hamda ko‘p o‘rinda bunday yondoshuvni rag‘batlantiradi. Bunda dastur kodining murakkabligi ancha kamayadi. Hosila sinf (avlod) o‘z bazaviy sinfining (otasining) hamda sinflar tabaqalanishidagi o‘zining barcha ajdodlarining hamma xususiyatlari, metodlari va voqealarini voris qilib oladi.

Sodda vorislik

Vorislik paytida bazaviy sinf yangi atributlar va operatsiyalar hisobiga yanada o‘sadi. Hosila sinfda odatda yangi ma’lumotlar a’zolari, xususiyatlar va metodlar paydo bo‘ladi. Obyektlar bilan ishlashda dasturchi odatda aniq masalani hal qilish uchun eng to‘g‘ri keladigan sinfni tanlaydi, hamda undan bitta yoki bir nechta voris avlod yaratadiki, ular o‘z otalarida mavjud imkoniyatlardan ko‘proq imkoniyatga ega bo‘ladilar. Do‘stona funksiyalar hosila sinfga barcha tashqi sinflar ma’lumotlari a’zolariga kirish huquqini olish imkonini beradilar.

Bundan tashqari, voris qilib olinayotgan metodlardan, ularning bazaviy sinfdagi ishi avlodga to‘g‘ri kelmasa, hosila sinf ortiqcha yuklanishi mumkin. OMD da ortiqcha yuklanishdan foydalanish har qanaqasiga rag‘batlantiriladi, garchi bu so‘zning to‘g‘ri ma’nosidan kelib chiqqanda, odatda ortiqcha yuklanishlar tavsiya qilinmaydi. Agar metod bittadan ortiq bir nomdagi funksiya bilan assotsiyatsiyalansa, u ortiqcha yuklangan deb aytiladi. E’tibor bering, sinflar tabaqalanishida ortiqcha yuklatilgan metodlarni chaqirib olib mexanizmi qayta aniqlangan funksiyalarni chaqirib olishdan mutlaqo farq qiladi. Ortiqcha yuklanish va qayta aniqlanish – bu turli tushunchalar. Virtual metodlar bazaviy sinf funksiyalarini qayta aniqlash uchun qo‘llanadi.

Vorislik kontsepttsiyasi

Vorislik kontsepttsiyasini soat haqidagi misolga tadbiq qilish uchun faraz qilaylikki, vorislik tamoyiliga amal qilgan «Casio» firmasi soatning yangi modelini

chiqarishga qaror qildi. Aytaylik, bu model, tugmachalardan biri ikki marta bosilsa, vaqtni ovozda ayta oladi. Gapiradigan soatlar modeli (OMD atamalari bo‘yicha, yangi sinf) ni yangidan yaratish o‘rniga muhandislar ishni uning prototipidan boshlaydilar (OMD atamalari bo‘yicha, bazaviy sinfning yangi avlodini yaratadilar). Hosila obyekt otasining barcha atributlari va funktsionalligini voris qilib oladi. Sintezlangan ovozda aytilgan sonlar avlodning yangi ma’lumotlar a’zolari bo‘lib qoladi, tugmachalarning ob’ktli metodlari esa, ularning qo‘shimcha funktsionalligini ishga tushirish uchun, ortiqcha yuklatilgan bo‘lishi kerak. Tugmachalarning ikki marta bosilish hodisasiga yangi usul javob berib, u joriy vaqtga mos keladigan sonlar ketma-ketligi (yangi ma’lumotlar a’zolari) ning talaffuz qilinishida namoyon bo‘ladi. Yuqorida aytilganlarning hammasi gapiradigan soatlarning dasturiy amalga oshirilishiga to‘liq taalluqli.

Vorislikda murojaat huquqlarining boshqarilishi

Vorislik o‘zining barcha ajdodlarining xususiyatlari, ma’lumotlari, metodlari va voqealarini meros qilib oladigan xosila sinfini e’lon qilish imkoniyatini beradi, shuningdek yangi tavsiflarni e’lon qilishi hamda meros sifatida olinayotgan ayrim funksiyalarni ortiqcha yuklashi mumkin. Bazaviy sinfning ko‘rsatib o‘tilgan tavsiflarini meros qilib olib, yangi tug‘ilgan sinfni ushbu tavsiflarni kengaytirish, toraytirish, o‘zgartirish, yo‘q qilish yoki o‘zgarishsiz qoldirishga majburlash mumkin.
Xosila sinfni e’lon qilishning umumlashgan sintaksisi:
class <sinf nomi>: [<kirish huquqini beruvchi setsifikator >] <ajdod sinf nomi>
{…}
Sinf o‘zining bazaviy sinfidan yuzaga kelayotganida, uning barcha nomlari xosila sinfda avtomatik tarzda yashirin private bo‘lib qoladi. Ammo uni, bazaviy sinfning quyidagi kirish spetsifikatorlarini ko‘rsatgan holda, osongina o‘zgartirish mumkin:
private. Bazaviy sinfning meros bo‘lib o‘tayotgan (ya’ni ximoyalangan va ommaviy) nomlari xosila sinf nusxalarida kirib bo‘lmaydigan bo‘lib qoladi.

public. Bazaviy sinf va uning ajdodlarining nomlari xosila sinf nusxalarida qirib bo‘ladigan bo‘ladi, barcha ximoyalangan nomlar esa ximoyalangan bo‘lib qolaveradi.

Agarda yangi sinf class kalitli so‘z yordamida aniqlangan bo‘lsa unda xosila sinfdagi meros komponentalar private kirish statusiga ega bo‘ladi, struct yordamida esa public statusiga.
Meroslikda ko‘rsatilmagan kirish statusini asosiy(bazaviy) sinf ismini oldidan ko‘rsatilgan private, protected va public kirish atributlari yordamida o‘zgartirish mumkin.

Virtual funksiyalar va abstrakt sinflar

Virtual funksiyalar mexanizmiga biror komponent funksiyaning har bir xosilaviy sinfda aloxida varianti mavjud bщlish lozim bщlganda murojaat qilinadi.

Bunday funksiyalarga ega sinflar polimorf sinflar deb ataladi va obyektli dasturlashda axloxida o‘ringa ega.
Virtual funksiyalar kechki yoki dinamik bog‘lanish mexanizmi asoslangandir. Asos sinf har qanday nostatik komponent funksiyasi virtual kalit so‘zi yordamida virtual deb e’lon qilinishi mumkin. Kechki bog‘lanishda erta bog‘lanishga o‘xshab adreslar statik ravishda kompilyatsiya jaryonida emas, balkim dinamik dastur bajarilishi jarayonida aniqlanadi. Bog‘lash jarayoni virtual funksiyalarni adreslar bilan almashtirishdan iborat. Virtual funksiyalar adreslar xaqida ma’lumot saqlanuvchi jadvaldan foydalanadi.
Virtuallik vorislikka o‘tadi. Funksiya virtual deb e’lon qilingandan so‘ng xosila sinfda qayta ta’rifi(shu prototip bilan) bu sinfda yangi virtual funksiyani yaratadi, bu holda virtual spetsifikatori talab qilinmaydi.
Destruktorlardan farqli konstruktorlar virtual bo‘lolmaydi. Amaliy jixatdan virtual funksiyaga ega har bir sinf virtual destruktorga ega bo‘lishi kerak.
Abstrakt sinflarning mexanizmi keyinchalik konkretizatsiyalanadigan umumiy tushunchalarni tavsiflash uchun ishlab chiqilgan. Bu holda, sinflar iyerarxiyasini yaratish quyidagi sxema bo‘yicha bajariladi.
Ierarxiya asosida abstrakt bazoviy sinf turadi. U interfeysni meros kilib olish uchun foydalaniladi. Xosila sinflar bu interfeysni konkretizatsiyalaydi va amalga oshiradi. Abstrakt sinfda sof virtual funksiyalar elon etilgan, ular aslida abstrakt usullar.
Ba’zi sinflar masalan shape sinfi, abstrakt tushunchalarni ifodalaydi va ular uchun obyekt yaratib bo‘lmaydi. Bunday sinflar biror xoila sinfda ma’noga ega bzladm:

Foydalanilgan adabiyotlar

1. Informatika va informatsion texnologiyalar, M. Aripov va boshqalar. Oliy o’quv yurti talabalari uchun darslik. Toshkent-2019 y.
2. Axborot texnologiyalari, M. Aripov va boshqalar. Oliy o’quv yurti talabalari uchun o’quv qo’llanma. Toshkent-2019 y.
3. Delphi tilida dasturlash asoslari, Sh. Nazirov. Toshkent-2018 y.