O‘ZBEKISTON RESPUBLIKASI OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI

TOSHKENT DAVLAT IQTISODIYOT UNIVERSITETI

O‘.T. XAYITMATOV, R.X. ALIMOV, A.A. AKRAMOV, O.X. AZAMATOV

OBYEKTGA YO‘NALTIRILGAN DASTURLASH TILLARI

O‘zbekiston Respublikasi Oliy va o‘rta maxsus ta’lim vazirligi huzuridagi Muvofiqlashtiruvchi kengash tomonidan 5111000 - Kasb ta’limi (informatika va axborot texnologiyalari) bakalavriat ta’lim yo‘nalishida tahsil olayotgan talabalar uchun o‘quv qo‘llanma sifatida tavsiya etilgan

TOSHKENT – «IQTISODIYOT» – 2019

UDK: 6P2.15.33
KBK:

Xayitmatov O‘.T., Alimov R.X., Akramov A.A., Azamatov O.X. Obyektga yo‘naltirilgan dasturlash tillari. O‘quv qo‘llanma. - T.: **«IQTISODIYOT», 2019. – 139 bet.**

O‘quv qo‘llanma obyektga yo‘naltirilgan dasturlash tillari tasnifi, ularning tuzilishi, tarixi va rivojlanish tendentsiyasi, obyektli dasturlash asoslari, sinflar va sinflarda vorislik kabi masalalarni o‘z ichiga qamrab olgan. Shuningdek, murakkab ma’lumotlar turlari, sinflar va obyektlar, obyektlar massivlari, sinflar va ko‘rsatkichlar, sinflar orasida munosabatlar, standart amallarni qo‘shimcha yuklash, funksiyalar va sinflar shablonlari, standart shablon sinflar bibliotekasi, xodisalarni qayta ishlash, loyihalash xamda zamonaviy dasturlash tillari to‘g‘risida ma’lumotlar keltirilgan.

Mazkur o‘quv qo‘llanma informatika va axbotot texnologiyalari yo‘nalishda ta’lim olayotgan talabalar uchun mo‘ljallangan.

Учебник охватывает такие темы, как классификация объектно-ориентированных языков программирования, их структура, история и тенденции развития, основы объектно-ориентированного программирования, классная комната и наследование классов. Он также включает сложные типы данных, классы и объекты, массивы объектов, классы и индикаторы, отношения между классами, добавление стандартных действий, функций и шаблонов классов, стандартную библиотеку классов шаблонов, обработку событий, информация о дизайне и современных языках программирования.

Учебное пособие предназначено для студентов в области информатики и информационных технологий.

The textbook covers topics such as classification of object-oriented programming languages, their structure, history and development trends, the basics of object-oriented programming, classroom and class inheritance. It also includes complex data types, classes and objects, arrays of objects, classes and indicators, relationship between classes, addition of standard actions, functions and class templates, standard template class library, event processing, information on design and modern programming languages.

This textbook is intended for students in the field of computer science and information technology.

Taqrizchilar:
F. Agzamov - i.f.n., dots. TATU o‘quv ishlari bo‘yicha prorektor
Sh.X.Xoshimxodjayev- i.f.n., dots. TDIU “Iqtisodiyotda axborot texnologiyalari” kafedrasi dotsenti

ISBN: 000000000

UO‘K: 6P2.15.33
KBK 00000

KIRISH

O‘zbekiston Respublikasi Prezidenti Shavkat Mirziyoyev O‘zbekiston Respublikasi Konstitutsiyasi qabul qilinganining 24 yilligiga bag‘ishlangan tantanali marosimdagi ma’ruzasida “Biz ta’lim va tarbiya tizimining barcha bo‘g‘inlari faoliyatini bugungi zamon talablari asosida takomillashtirishni o‘zimizning birinchi darajali vazifamiz deb bilamiz”, - deb ta’kidlagan edilar.

Kelajagi buyuk davlatni qurish tafakkuri, dunyoqarashi o‘zgargan xodimlarimiz, mutaxassislarimizga ko‘p jixatdan bog‘liqdir. Yangicha fikrlaydigan, bozor sharoitlarida muvaffaqiyatli xo‘jalik yuritadigan, yuksak malakali, chuqur bilimli mutaxassislarni tayyorlash davr talabi bo‘lib qoldi. Mamlakatimiz rivojlangan davlatlar qatoridan mustaxkam o‘rin egallashi uchun zamonaviy kompyuter texnologiyalarini xayotimizning barcha jabxalariga keng joriy etib kelinmoqda.

Xozirgi vaqtda kompyuter texnologiyalari rivojlanib borayotgan vaqtda asosiy e`tibor foydalanuvchilarga emas, dasturchilarga qaratilmoqda. Shuni hisobga olib, talabalarda dasturlar tuzish ko‘nikmalarini hosil qilish va ularda kerakli amaliy dasturlarni yarata olish imkoniyati shakllantirishga katta e`tibor berilmoqda.

O‘zbekiston Respublikasi Kadrlar Tayyorlash Milliy dasturida ham yosh avlodni yangi kompyuter texnologiyalarini egallashi, ularning bilimlarini chet el talabalari saviyasiga etkazish dolzarb masala qilib qo‘yilgan.

So‘ngi yillarda kompyuterning dasturiy ta’minoti rivojlanishi asosiy yo‘nalishlaridan biri bu obyektga yo‘naltirilgan dasturlash sohasi bo‘ldi. Obektga yo‘naltirilgan operatsion tizimlar (Masalan, Windows), amaliy dasturlar va obyektga yo‘naltirilgan dasturlash tizimlari ham ommaviylashdi.

Birinchi obyektga yo‘naltirilgan dasturlash elementi Simula-67 (1967 y., Norvegiya) tili bo‘ldi. Turbo PASCAL ning 5,5 versiyasidan boshlab obyektga yo‘naltirilgan dasturlash vositalari paydo bo‘ldi. Turbo PASCAL ning rivoji yakuni sifatida BORLAND filmasi tomonidan DELPHI dasturlash tizimi yaratilishi bo‘ldi. Ushbu tizim yordamida tez va oson murakkab bo‘lgan grafik interfeysni

dasturlash imkoniyati mavjud. 1991- yilda Visual BASIC ning I versiyasidan boshlab bu til to‘laligicha obyektga yo‘naltirildi (1997- yil).

Obyektga yo‘naltirilgan dasturlash tillaridan yana biri 1995- yilda Jeyms Gosling boshchiligida Sun Microsystems kompaniyasida yaratilgan JAVA tilidir. Uni ishlab chiqishda maxsus o‘rganish talab qilmaydigan, sodda tilni maqsad qilingan.

JAVA tili maksimal darajada C ++ tiliga o‘xshash bo‘lishi uchun yaratilgan. JAVA Internet uchun dasturlar tayyorlashning ideal vositasidir. So‘ngi yillarda Microsoft kompaniyasi tomonidan C++ davomchisi sifatida C # (Ci sharp) tili yaratildi.

1985- yilda Bell Labs (AQSh) laboratoriyasi C++ dasturlash tili yaratilganligini xabarini berdi. Bugungi kunda bu til obyektga yo‘naltirilgan dasturlash tillari orasida mashhurdir. Bu til yordamida istalgan mashina uchun – shaxsiydan to superkompyuterlargacha dasturlar yozish mumkin. Bu tilning asoschisi Born Straustrupdir.

1 bob. DELPHI VIZUAL DASTURLASH MUHITI HAQIDA ASOSIY TUSHUNCHALAR

1.1. Delphi dasturlash muhiti

Delphi - Windows operatsion tizimida dastur yaratishga yo‘naltirilgan dasturlash muhitidir. Delphida dastur tuzish zamonaviy vizual loyihalash texnologiyalariga asoslangan bo‘lib, unda dasturlashning obyektga mo‘ljallangan g‘oyasi mujassamlashgan. Delphida dastur Turbo Pascal dasturlash tilining rivoji bo‘lgan Object Pascal tilida yoziladi.

Delphi - bir necha muhim ahamiyatga ega bo‘lgan texnologiyalar kombinatsiyasini o‘zida mujassam etgan:

- yuqori darajadagi mashinali kodda tuzilgan kompilyator;
- obyektga mo‘ljallangan komponentalar modullari;
- dastur ilovalarini vizual tuzish;
- ma’lumotlar bazasini tuzish uchun yuqori masshtabli vosita.

Delphi - Windows muhitida ishlaydigan dastur tuzish uchun qulay bo‘lgan vosita bo‘lib, kompyuterda dastur yaratish ishlarini avto- matlashtiradi, xatoliklarni kamaytiradi va dastur tuzuvchi mehnatini yengillashtiradi. Delphida dastur zamonaviy vizual loyihalash texnolo- giyasi asosida obyektga mo‘ljallangan dasturlash nazariyasini hisobga olgan holda tuziladi. Delphi tizimi Turbo Pascal 7.0. tilining rivoji bo‘lgan obyektga mo‘ljallangan Object Pascal dasturlash tilini ishlatadi.

Ma’lumki, dastur tuzish sermashaqqat jarayon, lekin Delphi tizimi bu ishni sezilarli darajada soddalashtiradi va masala turiga qarab dastur tuzuvchi ishining 50—80%ni tizimga yuklaydi. Delphi tizimi dasturni loyihalash va yaratish vaqtini kamaytiradi, hamda Windows muhitida ishlovchi dastur ilovalarini tuzish jarayonini osonlashtiradi.

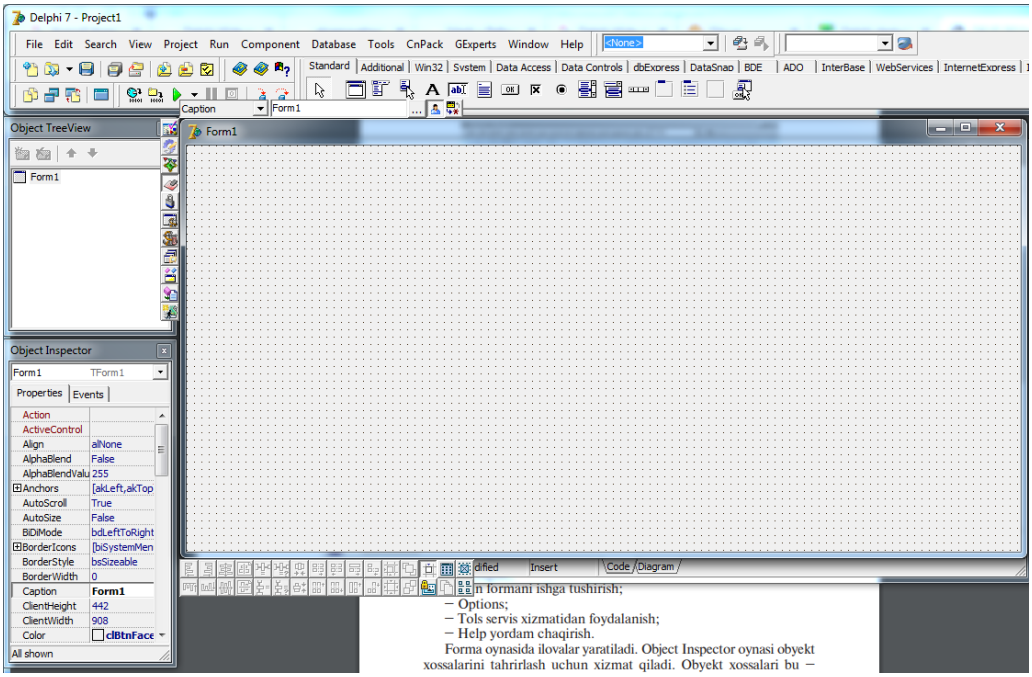
Delphi o‘zida bir qancha zamonaviy ma’lumotlar bazasini boshqarish tizimlarini va dasturlash texnologiyalarini ma’lumotlar bazasini yaratishda ishlatadi.

1.2. Delphi tizimining oynasi va uning elementlari

Delphi tizimida ishni boshlash uchun uni dasturlar menyusidan topib ishga tushiramiz.

Пуск=>Программы=>Borland Delphi=>Delphi

Delphi oynasi koʻrinishi odatdagidan ancha boshqacharoq boʻlib, u oʻz ichiga beshta oynani oladi (1 – rasm):



1 – rasm. Delphi tizimining oynasi

- bosh oyna — Delphi Projectl;
- forma oynasi — Forml;
- obyekt xossalarini tahrirlash oynasi —Object Inspector;
- obyektlar roʻyxatini koʻrish oynasi — Object tree View;
- dastur kodlarini tahrirlash oynasi — Unit.pas.

Bosh oyna ekranning yuqori qismida joylashgan boʻlib, uning birinchi qatorida sarlavha, ya'ni, proyektning nomi joylashgan. Ikkinchi qatorda buyruqlar menyusi gorizontal koʻrinishda joylashgan. Keyingi qatorning chap tarafida uskunalar paneli va oʻng tarafida komponentalar politrasi joylashgan.

Buyruqlar menyusi quyidagilarni oʻz ichiga olgan:

— File (fayl) boʻlimi fayllar ustida ish bajarish uchun kerakli buyruqlarni

oʻz ichiga olgan;

— Edit (tahrir) boʻlimi fayl ichidagi maʼlumotlarni tahrirlash uchun kerakli buyruqlarni oʻz ichiga olgan:

— Seerch;

— View;

— Compile;

— Run formani ishga tushirish;

— Options;

— Tols servis xizmatidan foydalanish;

— Help yordam chaqirish.

Forma oynasida ilovalar yaratiladi. Object Inspector oynasi obyekt xossalarini tahrirlash uchun xizmat qiladi. Obyekt xossalari bu — obyektga berilgan xarakteristika boʻlib, uning koʻrinishi, joylashishi va holatidir. Masalan, Width va Height xossalari forma oʻlchamini, top va Lift esa formaning ekrandagi holati, Caption — sarlavha matnini aniqlaydi.

Vizual dasturlash texnologiyasida obyekt deganda muloqot oynasi va boshqarish elementlari (kiritish va chiqarish maydoni, buyruq tugmalari, pereklyuchatellar va boshqa) tushuniladi.

Delphida dasturlash ikkita oʻzaro taʼsir etuvchi bir-biri bilan bogʻliq jarayon asosida tashkil qilinadi:

— dasturni vizual loyihalash jarayoni;

— dastur kodlarini kiritish (yozish) jarayoni.

Kodlarni yozish uchun maxsus kod oynasi mavjud boʻlib, u dastur matnini kiritish va tahrirlash uchun moʻljallangandir. Bu kodlarni yozish oynasida dasturlash Pascal tilining rivoji boʻlgan va kengaytirilgan Object Pascal tilida tuziladi.

Kodlarni yozish oynasi boshlanishda oʻz ichiga holi boʻsh formani akslantiruvchi dastur matnini yozib chiqaradi. Dastur loyihasini ishlash mobaynida dasturchi kerakli dastur operatorlarini kiritib, formani loyiha boʻyicha akslantiradi. Delphida dasturlash forma oynasini tashkil etishdan boshlanadi.

Oddiy dastur ilovasini yaratish ketma-ket **File=> New=> Application** buyrug‘ini berish bilan boshlanadi. Bu buyruqni berishdan oldin ikkita asosiy ishni bajarish lozim:

— papka tashkil etish;

— tizimni to‘g‘rilash.

Delphi dasturlash muhitida ishlash jarayonida quyidagi kengaytmali fayllar ishlatiladi:

— loyiha fayli, kengaytmasi **.dpr**;

— paskal moduli fayli, kengaytmasi **.pas**;

— komponentalar joylashgan fayl, kengaytmasi **.dcu**;

— formalar joylashgan fayl, kengaytmasi **.dfm**;

— ma’lumotlar bazasi fayli, kengaytmasi **.dbf**.

Tayyorlanadigan Delphi dastur uchta asosiy etapdan o‘tadi:

— kompilyatsiya;

— komponovka;

— bajarish.

Kompilyatsiya etapida tayyorlangan dastur matni Object Pascal tiliga o‘tkaziladi. Kompanovka bosqichida esa kerakli qo‘shimcha yordamchi dasturlar va ostdasturlar unga birlashtiriladi. **F9** tugmasini bosish bilan **Save UnitAs** dialog oynasi paydo bo‘ladi va sizdan **Unit.pas** moduli uchun fayl nomini va joylashadigan papkani ko‘rsatishingizni so‘raydi. Agar joyi ko‘rsatilmasa Delphi avtomatik ravishda dasturingizni **Bin** papkasiga joylashtiradi. Yaxshisi siz bu papkani o‘z ishchi papkangiz nomiga almashtiring, masalan, **My_Delphi**. Dastur kompilyatsiya qilinishi paytida Delphi sistemasi **pas**, **dfm** va **dcu** kengaytmali modullar tuzadi. **.pas** kengaytmali fayl kodlarni yozish oynasiga kiritilgan dastur matnini, **.dfm** forma oynasi tashkil etuvchilarini, **.dcu** kengaytmali fayl esa **.pas** va **.dfm** kengaytmali fayllarning birgalikdagi mashina kodiga o‘tkazilgan variantini saqlaydi. Bu **.dcu** kengaytmali fayl kompilyator tomonidan tashkil qilinadi va yagona ishchi (bajariluvchi) **.exe** kengaytmali fayl tashkil qilishga baza yaratadi.

1.3. Delphi loyihasining tuzilmasi

Delphi dasturi — bu bir necha bir-biri bilan bog‘liq fayllardir. Har qanday dastur .dpr kengaytmali loyiha fayli va bir yoki bir necha .pas kengaytmali modullardan tashkil topadi. Loyiha fayli dasturchi tomonidan kiritilmaydi, u foydalanuvchining ko‘rsatmalari asosida avtomatik ravishda Delphi sistemali dasturi tomonidan tuziladi. Loyiha fayli matnini ko‘rish uchun Project/View Source buyrug‘ini berishi zarur. Loyiha matni umumiy holda quyidagicha bo‘lishi mumkin:

Program Projectl;
Uses
Forms,
Unitl in ‘Unitl.pas’ {Forml}
{$R *.res}
Begin
Application.Initialize;
Application.CreateForm(Tform1, Forml);
Application.Run;
End.

Loyiha nomi dasturchi tomonidan loyiha faylini saqlash vaqtida beriladi va u Delphi muhitida bajariluvchi fayl, ya’ni, kengaytmasi .exe bo‘lgan faylni tashkil qilishni aniqlaydi. Loyiha faylidan keyin ishlatiladigan modullar: standart modullar Forms va Unitl joylashadi. **{$R *.res}** direktivasi kompilyatorga ishlatilishi kerak bo‘lgan resurs fayllari, masalan dasturlarni e’lon qilish kerakligini bildiradi. Yulduzcha belgisi resurs faylining kengaytmasi .res ekanligini bildiradi. Bosh modulning bajariluvchi qismi **Begin .. End** operatorlari orasiga joylashadi.

Modul — bu, biror-bir dastur. Modullar standart konstruksiyasiga ega. Object Pascalda modul tuzilmasi umumiy holda quyidagi ko‘rinishda bo‘ladi:

Unit <Modul nomi>
Interfase
…….
Implementation

……..

Initialization

……..

Finalization

……..

End.

Delphi tizimini ishga tushirgandan keyin modul tuzilmasi quyidagi ko‘rinishda bo‘ladi:

Unit unit1;

Interface

Uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs;

Type

TForm1 = class(TForm)

Private

{ Private declarations }

Public

{ Public declarations }

end;

Var

Form1: TForm1;

Implementation

{$R *.dfm}

End.

1.4. O‘zgaruvchilar va konstantalar

O‘zgaruvchilar. O‘zgaruvchi nomi ostiga chizish belgisi yoki lotin harfidan boshlanuvchi lotin harflari, arab raqamlari va ostiga chizish belgilari ketma ketligi ya’ni identifikatordir.

O‘zgaruvchilarning quyidagi tiplari mavjuddir: **char** (simvol), **short** (qisqa butun), **int** (butun), **long** (uzun butun), **float** (haqiqiy), **double** (ikkilangan haqiqiy).

Butun sonlar ta’riflanganda qurilgan tiplar oldiga **unsigned** (ishorasiz) ta’rifi qo‘shilishi mumkin. Ishorali ya’ni **signed** tipidagi sonlarning eng katta razryadi son ishorasini ko‘rsatish uchun ishlatilsa **unsigned** (ishorasiz) tipdagi sonlarda bu razryad sonni tasvirlash uchun ishlatiladi

O‘zgaruvchilar ta’rifi sodda shakli:

<tip> <o‘zgaruvchilar_nomlari_ro‘yxati >;

o‘zgaruvchilarni ta’riflashda boshlangich qiymatlarini ko‘rsatish mumkin.

<tip> <o‘zgaruvchilar_nomlari_ro‘yxati >= <initsializator>;

Bu usul initsializatsiya deyiladi.

Misollar:

float pi = 3.14 , cc=1.3456;

unsigned int year = 1999;

Konstantalar. Konstanta bu o‘zgartirish mumkin bo‘lmagan qiymatdir:

Butun. O‘nlik – oldida 0 emas (masalan: 8, 0, 192345); sakkizlik – oldida 0 (masalan: 016, 01); o‘n oltilik – oldida 0x yoki 0X (masalan: 0xA, 0X00F);

Simvolli. Oddiy - ‘g‘’,’r,’6’; boshqaruvchi –‘\n’, ‘\0xF5’;

Haqiqiy. Fiksirlangan nuqtali (masalan: 5.7, .0001, 41.); suzuvchi nuqtali (masalan: 0.5e5, .11e-5, 5E3) ;

Mantiqiy. Mantiqiy konstantalar **true** (rost) va **false** (yolg‘on). Ichki ko‘rinishi false – 0, ixtiyoriy boshqa qiymat true deb qaraladi.

Nomlangan konstantalar. Nomlangan konstantalar quyidagi shaklda kiritiladi:

Const tip konstanta_nomi=konstanta_qiymati.

Ko‘zda tutilgan bo‘yicha int tipidan foydalaniladi.

Misol uchun:

Const double EULER=2.718282;

Const long M=99999999;

Const R=765;

Yangi tip. Typedef ta’riflovchisi yangi tiplarni kiritishga imkon beradi.Misol uchun yangi COD tipini kiritish:

Typedef unsigned char COD;

COD simbol;

1.5. Amallar

Arifmetik amallar. Binar amallar: qo‘shish (+),ayirish (–),ko‘paytirish (*), bo‘lish (/) va modul (%) olish.

Misol uchun **20/3=6; (-20)/3=-6; 5%2=1;**

Unar amallar: unar minus – va unar +; inkrement ++ va dekrement--. Inkrement va dekrement prefiks ya’ni ++i, postfiks ya’ni i++ ko‘rinishda ishlatishi mumkin. Masalan agar **i=2**, u holda **3+(++i)= 6**, **3+i++=5** ga teng bo‘ladi. Ikkala holda ham **i=3** ga teng bo‘ladi.

Nisbat amallari. Nisbat amallari qiymatlari quyidagilar: mantiqiy katta (>); kichik (<); katta yoki teng (>=); kichik yoki teng (<=); teng (==) ; teng emas (!=).

Mantiqiy amallar. Mantiqiy amallari qiymatlari mantiqiy bo‘ladi. Mantiqiy amallarda quyidagilardan foydalaniladu: **||** (diz’yunksiya), **&&** (kon’yunksiya), **!** (inkor).

Razryadli amallar. Razryadli amallar **|** (diz’yunksiya), **&** (kon’yunksiya), **^** (XOR), **!** (inkor). Masalan 5 kodi 101 ga teng va 6 kodi 110 ga teng:

6 & 5 = 4 = 100; 6 | 5 = 7 = 111; 6 ^ 5 = 3 = 011; ~6 = 4 = 010.

Chapga surish << amali.Masalan:5 << 2 = 20 yoki 101 << 2 = 10100.

O‘ngga surish >> amali. Masalan 5 >> 2 = 1 yoki 101 >> 2 = 001 = 1.

Qiymat berish amali. Oddiy qiymat berish amali:

O‘zgaruvchi_nomi = ifoda;

Z = 4.7 + 3.34; C = y = f = 4.2 + 2.8;

Murakkab qiymat berish amali:

O‘zgaruvchi_nomi amal= ifoda;

Bu yerda amal quyidagi amallardan biridir: *****, /, **%**, +, – , **&**, ^, |, <<, >>.

Misol uchun: **x+ = 4** ifoda **x = x + 4** ifodaga ekvivalentdir;

x >> = 4 ifoda **x = x >> 4** ifodaga ekvivalentdir;

Shartli amal. Shartli amal ternar amal deyiladi:

<1-ifoda> ? <2-ifoda> : <3-ifoda>

Masalan:**a < b ? a : b**.

Tiplar bilan ishlovchi amallar. Tiplarni o‘zgartirish amali quyidagi ikki kurinishga ega:

Kanonik: (tip_nomi) operand; masalan**: r=(unsigned long)1;**

Funksional: tip_nomi (operand); masalan: **z=double(1);**

Xotiradagi hajmni hisoblash **sizeof** amalining ikki ko‘rinishi mavjud:

sizeof ifoda masalan: **Sizeof 3.14=8**

sizeof (tip) masalan: **Sizeof(char)=1**

1.6. Ko‘rsatkichlar va ilovalar

Ko‘rsatkichlar ta’rifi. Ko‘rsatkichlar qiymati konkret tipdagi obyektlar uchun xotirada ajratilgan adreslarga tengdir. Shuning uchun ko‘rsatkichlar ta’riflanganda ularning adreslarini ko‘rsatish shart. O‘zgaruvchi ko‘rsatkichlar quyidagicha ta’riflanadi.

<tip> * <ko‘rsatkich nomi>

Misol uchun **int * lp, lk** .

Ko‘rsatkichlarni ta’riflaganda insializatsiya qilish mumkindir. Initsializatsiya quyidagi shaklda amalga oshiriladi:

<tip> * <ko‘rsatkich nomi> = <konstanta ifoda>

Konstanta ifoda sifatida **&** simvoli yordamida aniqlangan obyekt adresi keladi. Misol uchun:

char c=’d’; char* pc=&c;

Ilovalar. Ilova biror obyektning o‘zgacha nomidir. Ilovalar quyidagicha ta’riflanishi mumkin:

<tip>& <Ilova_nomi>=<ifoda>

<tip>& <Ilova_nomi>(<ifoda>)

Misol uchun:

int l=777; int& rl=l; int& pl(l)

Rl yoki pl qiymatlarini o‘zgartirish avtomatik ravishda l ning ham qiymati o‘zgaradi.

Ko‘rsatkichlarga o‘xshab Ilovalarning qiymatlari ham adreslardir. Lekin Ilovalarning qiymatlarini o‘zgartirish mumkin emas va Ilovalarga murojaat qilinganda avtomatik ravishda * qiymat olish amali bajariladi.

Ilovalar bilan ishlash qoidalari. Ilova o‘zgaruvchi emasdir. Ilovaga bir marta qiymat bergandan so‘ng uni o‘zgartirish mumkin emas. Bundan tashqari ilovalar ustida quyidagi amallarni bajarish mumkin emasdir:

- Ilovaga ko‘rsatkich qiymatini berish mumkin emas.
- Ilovalarni solishtirish mumkin emas.
- Ilovalar ustida arifmetik amallar bajarish mumkin emas.
- Ilovani o‘zgartirish mumkin emas.

1.7. Foydalanuvchi funksiyalari

Funksiya ta’rifi. Funksiyani quyidagi ikki sifatda qarash mumkin:

- hosila tiplardan biri;
- dastur bajariluvchi minimal moduli.

Funksiya ta’rifi umumiy ko‘rinishi quyidagichadir:

<tip> <funksiya nomi>(<formal_parametrlar_ta’rifi>)

Formal parametrlarga ta’rif berilganda ularninga boshlangich qiymatlari ham kursatilishi mumkin.

Funksiya qaytaruvchi ifoda qiymati funksiya tanasida return <ifoda> ; operatori orqali ko‘rsatiladi. Misol:

float min(float, float b) { if (a<b) return a; return b; }

Funksiyaga murojaat qilish quyidagicha amalga oshiriladi:

<Funksiya nomi> (<haqiqiy parametrlar ro‘yxati>)

Masalan:

int x=5,y=6,z; z=min(x,y) yoki int z=min(5,6) yoki int x=5; int z=min(x,6)

Funksiya ta’rifida formal parametrlar initsializatsiya qilinishi, ya’ni boshlang‘ich qiymatlar ko‘rsatilishi mumkin.

Misol uchun:

float min(float a=0.0, float b=0) { if (a<b) return a; return b; }

Bu funksiyaga quyidagicha murojaat qilish mumkin:

int y=6,z; z=min(,y) yoki int z=min(,6);

Prototip. Agar programmada funksiya ta’rifi murojaatdan keyin berilsa, yoki funksiya boshqa faylda joylashgan bo‘lsa, murojjatdan oldin shu funksiyaning prototipi joylashgan bo‘lishi kerak. Prototip funksiya nomi va formal parametrlar tiplaridan iborat bo‘ladi. Formal parametrlar nomlarini berish shart emas.

Misol uchun:

float min(float, float);

Protseduralar. Funksiyaga parametrlar qiymat bo‘yicha uzatiladi. Funksiyaga parametrlar qiymatlari uzatilishi haqiqiy parametrlar qiymatlarini funksiya tanasida o‘zgartirish imkonini bermaydi. Bu muammoni hal qilish uchun ko‘rsatkichlardan foydalanish mumkin.

Misol:

```
void change (int &a, int &b)
{
int r;
r = a;  a = b; b = r;
}
// funksiya chaqirig‘i
change(a, b);
```

Funksiyalarni qo‘shimcha yuklash. Funksiyalarni qo‘shimcha yuklashdan maqsad bir xil nomli funksiyaga har xil tipli o‘zgaruvchilar bilan murojaat qilib qiymat olishdir. Kompilyator haqiqiy parametrlar ro‘yxati va funksiya chaqirig‘i asosida qaysi funksiyani chaqirish kerakligini o‘zi aniqlaydi.

Misol uchun har xil o‘zgaruvchilarni ko‘paytirish uchun quyidagi funksiyalar kiritilgan bo‘lsin:

float min(float a, float b) { if (a<b) return a; return b; }

int min(int a, int b) { if (a<b) return a; return b; }

Quyidagi murojaatlarning har biri to‘g‘ri bajariladi:

int m=min(6, x); float c=min(5, y);

Nazorat uchun savollar

1. Asosiy tiplarni ko‘rsating.
2. Ko‘rsatkich ta’rifini keltiring.
3. Ko‘rsatkichlar bilan bog‘liq amallarni keltiring
4. Ilova ko‘rsatkichdan qanday farq qiladi?
5. Satr simvolli massivdan qanday farq qiladi?
6. Massivlarni initsializatsiya qilish usullarini ko‘rsating.
7. Satrlarni initsializatsiya qilish usullarini ko‘rsating.
8. Qanday qilib massivlar formal parametrlar sifatida ishlatilishi mumkin?

2 bob. MASSIVLAR VA SATRLAR

2.1. Massiv va satr tushunchalari. Massivlarni e’lon qilish

Massiv indeksli o‘zgaruvchidir. Massiv sodda ta’rifi:

<tip> <o‘zgaruvchi_nomi>>[<konstanta_ifoda>] = <initsializator>;

Massiv indekslar qiymati har doim 0 dan boshlanadi. Ko‘p o‘lchovli massiv initsializatsiya qilinganda massivning birinchi indeksi chegarasi ko‘’rsatilishi shart emas, lekin kolgan indekslar chegaralari kursatilishi shart. Misol uchun:

```
int a[6]; float b[8],c[100];
double d[] = {1, 2, 3, 4, 5};
int A [20][10];
int A [30][20][10];
int A [3][3] = {0,1,2,3,4,5,6,7,8,9,10,11};
int A[ ][3] = { {0,1,100}, {200,210,300}, {1000, 2000, 2100}};
```

Satrlar. Satrli konstanta ikkilik qavslarga olingan simvollar ketma ketligidir. Satrli konstanta oxiriga avtomatik ravishda satr kuchirish ‘\n’ simvoli qo‘shiladi. Satr qiymati simvolli konstanta bo‘lgan simvolli massiv sifatida ta’riflanadi.

Misol uchun:

```
Char capital[]=”TASHKENT”;
Char capital[]={‘T’,’A’,’S’,’H’,’K’,’E’,’N’,’T’,’\n’};
Char A[ ][9] = { “Tashkent”, ”
```

Massivlar va satrlar funksiya parametrlari sifatida. Funksiyalarda massivlar argument sifatida ishlatilganda ularning birinchi indeksi chegarasini kursatish shart emas, qolganlarini chegarasini ko‘rsatish shart. Massivlar ilova bo‘yicha uzatiladi, ya’ni ularning qiymati funksiyada o‘zgarishi mumkin. Misol:

```
//massiv elementlari summasini hisoblash
int sum (int n, int a[] )
{  int i, int s=0;
for( i=0; i<n; i++ ) s+=a[i];
return s; }
```

Satrlar parametrlar sifatida char[] tipidagi bir o‘lchovli machssivlar sifatida uzatilishi mumkin. Bu holda satr uzunligini aniq ko‘rsatish shart emas.Misol:

```
//simvollar sonini hisoblash
int strlen ( char a[])
{ int i=0; while(a[i++]);
return i;}
```

2.2. Ko‘p o‘lchamli massivlar.

O‘zgaruvchi o‘lchamli massivlarni shakllantirish ko‘rsatkichlar va xotirani dinamik taqsimlash vositalari yordamida tashkil etiladi.

Xotirani dinamik taqsimlash uchun new va delete operatsiyalardan foydalaniladi. Operatsiya

new <tip_nomi> (<initsializator>)

tip ismi orqali belgilangan ma’lumotlar tipiga mos keluvchi o‘lchamli bo‘sh xotira qismini ajratish va unga murojaat etish imkonini beradi. Ajratilgan xotira qismiga initsializator orqali aniqlangan qiymat kiritiladi. Xotira ajratilsa xotira ajratilgan qismining bosh adresi qaytariladi, agarda xotira ajratilmasa NULL qaytariladi.

new operatsiyasi orqali oldindan ajratilgan xotira qismi delete operatsiyasi yordamida bo‘shatiladi.

Misollar:

```
int *i;  i=new int(10);
delete i;
```

Operatsiya

new <tip_nomi> (<initsializator>)

O‘zgaruvchilar massiviga xotira ajratishga imkon beradi.

Misollar:

```
int *mas=new[5];
delete [] mas;
```

Skalyar o‘zgaruvchilarga xotira ajratilish 1 misolda ko‘rsatilgan. Matritsani shakllantirishda oldin bir o‘lchovli massivlarga ko‘rsatuvchi ko‘rsatkich massivlar uchun xotira ajratiladi, keyin esa parametrli siklda bir o‘lchovli massivlarga xotira ajratiladi. Misol:

```
int n,m; cin>>n;
matr=new int*[n];
for (i=0;i<n;i++)
{cin>>m;
matr[i]=new int[m];
```

Xotirani bo‘shatish uchun bir o‘lchovli massivlarni bo‘shattiruvchi siklni bajarish zarur.

```
for(int i=0;i<n;i++)
delete matr[i];
```

keyin esa matr ko‘rsatkich ko‘rsatgan xotira bo‘shattiriladi.

```
delete [] matr;
```

2.3. Satrlar massiv sifatida

String tipi. Satrlar biln ishlash uchun standart bibliotekaga kiruvchi string murakkab turidan foydalanish qulaydir.

Bu tipdan foydalanish uchun quyidagi sarlavxali faylni ulash lozim:

- #include <string>
- Satrlarni ta’riflashga misollar:
- string st("BAXO \n"); //simvollar satri bilan initsiallash
- string st2; // bo‘sh satr
- string st3(st); shu tipdagi o‘zgaruvchi bilan initsiallash

Satrlar ustida amallar. Satrlar ustida quyidagi amallar aniqlangan:

- qiymat berish (=);
- ikki amal ekvivalentlikni tekshirish uchun (==) va (!=);
- konkatenatsiya yoki satrlarni ulash (+);
- qiymat berib qo‘shish amali (+=)

– indeks olish ([]).

Usullar: Satr uzunligini aniqlash uchun **size()** funksiyasidan foydalaniladi (uzunlik tugallovchi simvolni hisobga olmaydi).

cout << "uzunlik "<< st << ": " << st.size();

Maxsus **empty()** usuli agar satr bo‘sh bo‘lsa **true** qaytaradi, aks holda **false** qaytaradi:

if (st.empty()) // to‘g‘ri: bo‘sh

2.4. Satrlar bilan ishlovchi funksiyalar

Nusxa olishda satrlarni o‘zgartirish

String sinfi obyektlarini o‘zgartirish mumkin bo‘lmagani uchun xar gal satrni o‘zgartirganda StringBuffer obyektiga aylantirish, yoki bo‘lmasa String sinfining satr nusxasini yaratuvchi usullaridan foydalanish lozim.

substring

String sinfi obyektidan, **substring** usuli yordamida ostki satrni ajratib olish mumkin. Bu usul berilgan satr indekslari diapazonidan yangi simvollar nusxasini yaratadi.. Kerakli ostki satr faqat birinchi simoli indeksini ko‘rsatish mumkin. Bu holda ko‘rsatilgan indesdan oxirigacha hamma simvollardan nusxa olinadi. Bundan tashqari birinchi va oxirgi indeksni ko‘rsatish mumkin, bu holda ko‘rsatilgan indeksdan ikkinchi indeksgacha bo‘lgan hamma simvollardan nusxa olinadi.

"Hello World".substring(6) -> "World"

"Hello World".substring(3,8) -> "lo Wo"

concat

Satrlar konkatenatsiyasi ya’ni satrlarni ulash concat usuli yordamida amalga oshiriladi. Bu usul String sinfi yangi obyektini yaratib, berilgan satrni ko‘chiradi va usul parametrida ko‘rsatilgan satrni ulaydi.

"Hello".concat(" World") -> "Hello World"

replace

Simvollarni almashirish replace usuliga parametr sifatida ikki simvol uzatiladi. Satr yangi nusxasida birinchi simvolga mos simvollar ikkinchi simvolga almashtiriladi.

"Hello".replace('l' , 'w') -> "Hewwo"

toLowerCase i toUpperCase

Bu usullar juftligi hamma simvollarni quyi va yuqori registrga o‘tkazadi.

"Hello".toLowerCase() -> "hello"

"Hello".toUpperCase() -> "HELLO"

trim

Berilgan satr oldidagi va oxiridagi bo‘shlik simvollari trim usuli yordamida olib tashlanadi.

“Hello World “.trirn() -> "Hello World"

valueOf

Biror ma’lumotni matnli satrga aylantirish uchun valueOf usulidan foydalaniladi. Bu statik usul Java tilidagi hamma ma’lumotlar turlarida mavjuddir.

StringBuffer

StringBuffer — sinfida satrlar bilan ishlash uchun kerak bo‘lgan usullar mavjuddir. String sinfi obyektlari ma’lum uzunlikdagi o‘zgarmas satrlardan iborat. StringBuffer obyektlari uzunligii va qiymati o‘zgaruvchan satrlardan iborat. Java tilida ikkala sinfdan aktiv foydalaniladi, lekin dasturchilar + operatoridan foydalanilgan holda String sinfi obyektlarini ishlatadilar.

Konstruktorlar

StringBuffer sinfi obyektini parametrlarsiz yaratish mumkin, bu holda 16 simvol uchun joy ajratiladi va satr uzunligini o‘zgartirib bo‘lmaydi. Bundan tashqari konstruktorga bufer uzunligini ko‘rsatuvchi butun son berish mumkin. Bundan tashqari konstruktorga satr uzatish mumkin. Bu holda satrdan obyektga nusxa olinadi va qo‘shimcha 16 simvolga joy ajratiladi. StringBuffer obyekti uzunligini aniqlash uchun length usulidan, satr uchun ajratilgan joyni aniqlash uchun capacity usulidan foydalanish mumkin. Masalan:

```
class StringBufferDemo {
public static void main(String args[]) {
StringBuffer sb = new StringBuffer("Hello");
System.out.println("buffer = " + sb);
System.out.println("length = " + sb.length());
System.out. println("capacity = " + sb.capacity());
} }
```

Dastur natijasidan ko‘rish mumkinki String-Buffer obyektida qo‘shimcha joy ajratilgan.

```
S:\> java StringBufferDemo
buffer = Hello
length = 5
capacity = 21
```

ensureCapacity

StringBuffer obyekti yaratilgandan so‘ng yangi simvollarga joy ajratish uchun **ensureCapacity** usulidan foydalanish mumkin.

setLength

Buferdagi satr uzunligini o‘rnatish uchun setLength usulidan foydalanish mumkin. Agar obyektdagi satr uzunligidan katta qiymat berilsa, usul yangi satr oxirini nul kodli simvol bilan to‘ldiradi.

append

StringBuffer sinfining **append** usuli odatda ifodalarda + operatoridan foydalanilganda chaqiriladi. Xar bir parametr uchun String.valueOf usuli chaqiriladi va uning natijasi StringBuffer obyektiga ulanadi. Bundan tashqari append usuli StringBuffer obyektiga ilova qaytaradi. Bu esa misolda ko‘rsatilganidek chaqiriqlar ketma ketligini xosil qilishga imkon beradi.

```
class appendDemo {
public static void main(String args[]) {
String s;
int a = 42;
```

```
StringBuffer sb = new StringBuffer(40);
s = sb.append("a = ").append(a).append("!").toString();
System.out.println(s);
} }
```

Bu misol natijasi:

```
S:\> Java appendDemo
a = 42!
```

insert

Satrlarni ulash **insert** usuli append o‘xshash bo‘lib, xar bir ma’lumotlar tipi uchun bu usuning o‘z shakli bordir. Faqat append dan farqli String.valueOf usuli qaytargan simvollrni, StringBuffer obyekti oxiriga qo‘ymaydi, balki, birinchi parametri ko‘rsatgan, buferdagi joyga qo‘yadi. Keyingi misolimizda "there" satri "hello" va "world!" orasiga qo‘yiladi.

```
class insertDemo {
public static void main(String args[]) {
StringBuffer sb = new StringBuffer("hello world !");
sb.insert(6,"there ");
System.out.println(sb);
} }
```

Bu dastur bajarilganda quyidagi satr chiqadi:

```
S:\> java insertDemo
hello there world!
```

2.5. Satrlar sinf sifatida. Satrli sinflarni e’lon qilish

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. Masalan,

```
Type
  TchildClass=Class
     Fore: Integer;
     Ftwo: String;
     Fthree: Tobject;    End;
```

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. Masalan,

```
Var
 MyObject: TchildClass;
  Begin
MyObject.Fone:=16;
MyObject.Ftwo:=’qator qiymati’;
  End;
```

Maydon nomi unga mos xossa nomining birinchi harfi **“F”** bo‘lishi bilan farqlanadi.

Delphi dasturida qabul qilingan kelishuv bo‘yicha maydonlar nomlari **f** (**field** – maydon so‘zidan) harfidan boshlanishi lozim.

Nazorat uchun savollar

1. Sanovchi tip nima uchun qo‘llanadi?
2. Struktura elementlariga qanday murojaat qilinadi?
3. Bitli maydonlar qaysi tipga tegishli?
4. Birlashmalar asosiy xossalarini ko‘rsating.
5. Amallar new va delete nima uchun ishlatiladi?
6. Dinamik massivlarni xosil qilish usullarini ko‘rsating.

3 bob. MURAKKAB MA’LUMOTLAR TURLARI

3.2. Yangi ma’lumotlar turlarini kiritish

Agar har xil qiymatlarga ega bir nechta nomlangan konstanta kiritish kerak bo‘lsa sanovchi tipdan foydalanish mumkin:

enum <tip_nomi > {<konstantalar ro‘yxati >};

Konstantalar butun bo‘lishi kerak va initsializatsiya qilinishi mumkin, agar initsializator mavjud bo‘lmasa birinchi konstanta nol qiymat oladi, kolganlari bo‘lsa oldingisidan birga ortiq bo‘ladi.

Misol uchun:

enum{one=1,two=2,three=3};

Agar son qiymatlari ko‘rsatilmagan bulsa eng chapki so‘zga 0 qiymati berilib kolganlariga tartib buyicha usuvchi sonlar mos kuyiladi:

Enum{zero,one,two};

Bu misolda avtomatik ravishda konstantalar quyidagi qiymatlarni kabul kiladi:

Zero=0, one=1, two=2;

Konstantalar aralash ko‘rinishda kiritilishi ham mumkin:

Enum(zero,one,for=4,five,seeks}.

Bu misolda avtomatik ravishda konstantalar quyidagi qiymatlarni kabul kiladi:

Zero=0, one=1, for=4;five=5,seeks=6;

Nomlangan sanovchi tip kiritib, shu tipdagi o‘zgaruvchilar, ko‘rsatkichlar va ilovalardan foydalanish mumkin. Masalan:

enum color (black, green, yellow, blue, red, white);

color col=red;

color* cp=&col;

if (*cp==green) cout<<”yashil”;

3.2.Tuzilmalar va jamlanmalar

Struktura – bu ma’lumotlarni bir butun nomlangan elementlar to‘plamiga birlashtirish. Struktura elementlari (maydonlar) har xil tipda bo‘lishi mumkin va ular har xil nomlarga ega bo‘lishi kerak.

Strukturali tip quyidagicha aniqlanadi:

struct { <ta’riflar ro‘yxati > }

Strukturada albatta bitta komponenta bo‘lishi kerak. Struktura tipidagi o‘zgaruvchi quyidagicha ta’riflanadi:

<struktura_nomi > <o‘zgaruvchi>;

Struktura tipidagi o‘zgaruvchi ta’riflanganda initsializatsiya qilinishi mumkin:

<struktura_nomi > <o‘zgaruvchi>=<initsializator>;

Strukturani initsializatsiyalash uchun uning elementlar qiymatlarini figurali qavslarda tavsiflanadi.

Misollar:

1. struct Student

{

char name[20];

int kurs;

float rating;

};

Student s={”Qurbonov”,1,3.5};

2. struct

{

char name[20];

char title[30];

float rate;

}employee={“Ashurov", “direktor”,10000};

Strukturalarni o‘zlashtirish. Bitta tuzilma tipdagi o‘zgaruvchilar uchun o‘zlashtirish operatsiyasi aniqlangan. Bunda har bir elementdan nusxa olinadi. Masalan:

Student ss=s;

Struktura elementlariga murojaat. Struktura elementlariga murojaat aniqlangan ismlar yordamida bajariladi:

<Struktura_nomi>.<element_nomi>

Masalan:

employee.name – «Ashurov» satriga ko‘rsatkich;

employee.rate – 10000 qiymatga ega bo‘lgan butun tipdagi o‘zgaruvchi

Strukturaga ko‘rsatkichlar. Strukturaga ko‘rsatkichlar oddiy ko‘rsatkichlar kabi tasvirlanadi:

```
Student*ps;
```

Strukturaga ko‘rsatkich ta’riflanganda initsializatsiya kilinishi mumkin:

```
Student *ps=&mas[0];
```

Ko‘rsatkich orqali struktura elementlariga ikki usulda murojaat qilish mumkin. Birinchi usul adres buyicha qiymat olish amaliga asoslangan bo‘lib quyidagi shaklda qo‘llaniladi:

(* strukturaga ko‘rsatkich).element nomi;

Ikkinchi usul maxsus strelka (->) amaliga asoslangan bo‘lib quyidagi ko‘rinishga ega:

strukturaga ko‘rsatkich->element nomi

Struktura elementlariga quyidagi murojaatlar uzaro tengdir:

```
cin>>(*ps).name;
cin>>ps->title;
```

3.3. Tuzilmalar va massivlar

Strukturalarga yaqin tushuncha bu birlashma tushunchasidir. Birlashmalar union xizmatchi so‘zi yerdamida kiritiladi. Misol uchun:

union {long h; int i,j; char c[4]}UNI;

Birlashmalarning asosiy xususiyat shundaki uning hamma elementlari bir xil boshlang‘ich adresga ega bo‘ladi.

Masalan:

```
union{
char s[10];
int x;
}u1;
```

Quyidagi dastur yordamida bu xususiyatni tekshirish mumkin:

```
enum paytype{CARD,CHECK};
struct{
paytype ptype
union{
  char card[25];
  long check;
  };
}info;
switch (info.ptype)
{
case CARD:cout<<"\nKarta bilan to‘lash:"<<info.card;break;
case CHECK:cout<<"\nChek bilan to‘lash:"<<info.check;break;
}
```

3.4. Tuzilmalar va funksiyalar

Ma’lumotlar turlarini Delphi tilida umumiy holda ikkiga ajratish mumkin:

- standart turlar. Bu turlar oldindan Delphi tili tomonidan aniqlangan bo‘ladi;
- dasturchi tomonidan kiritiladigan (aniqlanadigan) turlar.

Standart turlar tarkibiga quyidagilar kiradi: butun, haqiqiy, belgili (simvol), qator (strok), mantiqiy, ko‘rsatgichli va variant.

Dasturchi turlarni dasturning **Var** bo‘limida o‘zgaruvchilarni tavsiflashda aniqlaydi yoki maxsus turlarni aniqlash uchun bo‘lim bo‘lgan -turlarni tavsiflash **Type** bo‘limida aniqlaydi.

Bu bo‘lim umumiy holda quyidagicha bo‘ladi.

```
  Type
        <tur nomi>=<turning tavsifi>;
```

Misol:

```
  Type
        TColor=(Red,Blue,Black);
    Var  Color1,Color2,Color3: TColor;
```

Type bo‘limida dasturchi tomonidan yangi TColor nomli tur kiritilmoqda va u Red,Blue,Black mumkin bo‘lgan qiymatlarni qabul qilishi mumkin.

Var bo‘limida dasturchi tomonidan turi aniqlangan uchta Color1,Color2,Color3 o‘zgaruvchilar tavsiflanmoqda.

Bu o‘zgaruvchilarni to‘g‘ridan to‘g‘ri quyidagicha ham tavsiflash mumkin.

Var Color1,Color2,Color3: (Red,Blue,Black);

Standart turlarni Type bo‘limida tavsiflash shart emas, ularni to‘g‘ridan to‘g‘ri Var bo‘limida tavsiflash mumkin.

Butun turlar. Butun turlar butun sonlarni tasvirlash uchun ishlatiladi. Jadvalda Delphi 7 da ishlatiladigan butun turlar ro‘yxati keltirilgan.

Tur	O‘zgarish diapazoni	O‘lcham (baytda)
Integer	-2147483648..2147483647	4
Cardinal	0..4294967295	4
SHortint	-128..127	1
Smallint	-32768..32767	2
Longint	-2147483648..2147483647	4
Int64	$-2^{63}..2^{63}-1$	8
Byte	0..255	1
Word	0..65535	2
LongWord	0..4294967295	4

Haqiqiy turlar. Haqiqiy turlar haqiqiy sonlarni tasvirlash uchun ishlatiladi. Jadvalda Delphi 7 da ishlatiladigan haqiqiy turlar ro‘yxati keltirilgan.

Tur	O‘zgrish diapazoni	O‘lcham (baytda)
Real	$5.0*10^{-324}..1.7*10^{308}$	8
Real48	$2.9*10^{-39}..1.7*10^{38}$	6
Single	$1.5*10^{-45}..3.4*10^{38}$	4
Double	$5.0*10^{-324}..1.7*10^{308}$	8
Extended	$3.6*10^{-4951}..1.1*10^{4932}$	10
Comp	$-2^{63}+1..2^{63}-1$	8

Belgili turlar. Ma’lumotlarning belgili turlari faqat bitta belgini saqlash uchun xizmat qiladi. Jadvalda Delphi 7 da ishlatiladigan belgili turlar ro‘yxati keltirilgan.

Tur	O‘lcham (baytda)
Char	1
ANSChar	1
WideChar	2

Mantiqiy turlar. Mantiqiy turlar chin (True) yoki yolg‘on (False) qiymatning birini qabul qiladi. Jadvalda Delphi 7 da ishlatiladigan mantiqiy turlar ro‘yxati keltirilgan.

Tur	O‘lcham (baytda)
Boolean	1
ByteBool	1
WordBool	2
LongBool	4

FloatToStr (n)	Haqiqiy n tasvirlovi satr
FloatToStrF(n, f , k,m)	Haqiqiy n tasvirlovi satr. Bunda: f - format; k - aniqlik; m - kasr qismidagi raqamlar soni
StrToInt (s)	Satrni butun songa o‘tkazish
StrToFloat (s)	Satrni haqiqiy songa o‘tkazish
Round (n)	Haqiqiy sonni yaxlitlash
Trunc (n)	Haqiqiy son kasr qismini olib tashlash
Frac(n)	Kasrli sonning kasr qismi
Int (n)	Kasr sonning butun qismi

Nazorat savollari:

1. Sanovchi tip nima uchun qo‘llanadi?
2. Struktura elementlariga qanday murojaat qilinadi?
3. Bitli maydonlar qaysi tipga tegishli?
4. Birlashmalar asosiy xossalarini ko‘rsating.
5. Amallar new va delete nima uchun ishlatiladi?
6. Dinamik massivlarni xosil qilish usullarini ko‘rsating.

4 bob. OBYEKTLI DASTURLASH ASOSLARI

4.1. 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:

- Obyektga mo‘ljallangan dasturlar yaratish;
- Obyektga mo‘ljallangan dasturlash tillari.

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.

4.2. 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;

sinf_nomi – identifikator;

sinf_komponentlari_ro‘yxati – sinfga tegishli ma’lumotlar va funksiyalar ta’rifi.

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>

4.3. 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 (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.

4.4. 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:

int nValuel=1; double fValue=nValuel+1.0;

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

int varl; int var2=1;varl=2*var2; nVariable= nVariable+2; nVariable+=2;

C++ da dastur tuzilmasi kuyidagi ko‘rinishga ega bo‘lishi mumkin:

// Eng sodda dastur

```
intmain() {
 return 0;
}
```

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.

C++ dadasturning bosh qismiga sarlavha fayllari kiritilishi lozim:

```
#include <iostream.h>
int main() {
 return 1;
}
```

Fayllar sarlavhasi yaratilishida chop qilish uchun sinf aniqlanadi va uning obyekti cout aniqlanadi:

1-misol

```
#include <iostream.h>
#include <math.h>
int main(){
        float x1,y1,x2;
        cout<<"Berilmalarnikiritish:\n";
        cin>>x1>>x2;
        x1=sin(x1);
        y1=cos(x2)+x1;
        cout<<"y1=  "<<y1<<"\n";
 return 0;}
```

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.

4.5. 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(maqolada Windows tizimi nazarda tutilgan):

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.

Nazorat uchun savollar

1. O‘zgaruvchilarning qanday turlari mavjud?
2. Mantiqiy o‘zgaruvchilar qanday qiymat qabul qiladi?
3. Qanday turlarni o‘zgartirish funksiyalari mavjud?
4. Delphi tilida dastur qanday strukturaga ega?
5. With instruksiyasi dastur tuzishda qanday imkoniyat yaratadi?
6. Yozuv qanday tur?
7. Dasturchi tomonidan kiritiluvchi turlar qanday turlar?

5 bob. SINFLAR VA OBYEKTLAR

5.1 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:

<sinf_tipi> <sinf_nomi>{<sinf_komponentlari_ro‘yxati>};

bu yerda:

sinf_tipi –class, struct, union xizmatchi so‘zlaridan biri;

sinf_nomi – identifikator;

sinf_komponentlari_ro‘yxati – sinfga tegishli ma’lumotlar va funksiyalar ta’rifi.

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>

5.2. 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 (protected) 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.

5.3 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 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.

Misol:

```
include<iostream.h>
class complex
{
 double re, im;                  // private ko‘zda tutilgan bo‘yicha
 public:
void show ();
somplex(double re1 = 0.0,double im1 = 0.0){re = re1; im = im1;}
};
inline void complex::show() { cout << "re=" << re<<"im=" << im;}
void main()
{ complex ss (5.9,0.15);
complex aa = complex (5.9,0.15);
  aa.show(); }
```

Konstruktor yordamida obyekt ma’lumotlarni initsiyalizatsiyalashni ikkita usuli mavjud:

Birinchi usulda parametrlar qiymatlari konstruktor tanasiga uzatiladi. Ikinchi usulda esa ushbu sinfdagi initsializatorlar ro‘yxatidan foydalanish nazarda tutilgan. Bu ro‘yxat parametrlar ro‘yxati va konstruktor tanasi orasiga joylashadi.

Sinfning biror obyekti uchun ajratilgan xotira obyekt yo‘qotilgandan so‘ng bo‘shatilishi lozimdir. Sinflarning maxsus komponentalari destruktorlar, bu vazifani avtomatik bajarish imkonini yaratadi. Destruktorni standart shakli quyidagicha:

~ <sinf_nomi> ( ) {<destruktor tanasi>}

Destruktor parametri yoki kaytariluvchi qiymatga ega bo‘lishi mumkin emas (xatto void tipidagi). Dastur obyektni o‘chirganda destruktor avtomatik chaqiriladi.

Agarda sinfda destruktor ochiq ko‘rsatilmagan bo‘lsa, unda kompilyator ko‘rsatilgan obyekt egallagan xotirani bo‘shatuvchi destruktorni generatsiyalaydi. Boshqa obyektlar egallagan xotirani bo‘shatmoqchi bo‘lsak, destruktorni ochiq aniqlash lozim. Masalan, string obyektdagi ch ko‘rsatgan saxifani

“Simvol satri” sinfi.

```
class string
{
  char *ch; // matnli satriga ko‘rsatkich
  int len;      // matnli satrni uzunligi
public:
  // konstruktorlar
  // bo‘sh satr – obyektni yaratish
  string(int N = 80): len(0){ch = new char[N+1]; ch[0] = ‘\0’;}
  // berilgan satr bo‘yicha obyekt yaratadi
  string(const char *arch){len = strlen(arch);
      ch = new char[len+1]; strcpy(ch,arch);}
  // funksiyalar – komponentlar
  // murojaatni satr uzunligiga qaytaradi
  int& len_str(void){return len;}
  // ko‘rsatkichni satrga qaytaradi
  char *str(void){return ch;}
```

```
//Nusxa olish klnstruktori
string(const string& st)
{len=strlen(st.len);
ch=new char[len+1];
strcpy(ch,st.ch);  }
//Destruktor
~string(){delete []ch;}
. . .};
```

5.4. Obyektlar massivlari. Sinf struktura kengaytmasi sifatida

Obyektlar aniqlangandan so‘ng shu obyektlarga ko‘rsatkichlar belgilash mumkin. Masalan:

```
complex A(5.2,2.7);
complex* PA=&A;
```

Obyektning umumiy elementlariga murojaat uchun -> operatsiyani yoki ism almashtirish va nuqta operatsiyasidan foydalanish mumkin

*PA.real() yoki PA->real;

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.

X sinfni ekvivalent ko‘rinishda shunday tavsiflash mumkin:

```
class x {
int m;
public:
int readm() { return this->m; }
};
```

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

Sinf kompanentasi yagona bo‘lib va hamma yaratilgan obyektlar uchun umumiy bulishi uchun uni statik element sifatida ta’riflash ya’ni static atributi orqali ta’riflash lozimdir. Obyektlarni yaratishda sinf statik ma’lumotlari takrorlanmaydi, ya’ni har bir statik komponentlar birdan-bir ko‘rinishda mavjud. Statik elementlarga murojat qilish uchun oldin initsializatsiya kilinishi lozim. Inizializatsiya quyidagicha amalga oshiriladi:

<Sinf-nomi>:: <kompleks-nomi> <initsializator>

Masalan:

int complex : : count = 0;

Bu taklifni sinfni aniqlashdan so‘ng global soxada joylashtirish lozim. Faqatgina sinf statistik ma’lumotlarini initsializatsiyalashda u xotiraga ega bo‘ladi va unga murojaat etish mumkin. Sinf statik ma’lumotlarga faqatgina obyekt ismi orqali murojaat etish mumkin. <obyekt_nomi>.<komponenta_nomi>

Masalan:

complex a; a.count=5;

Lekin, statik komponetlarga sinf obyekti aniqlanmagan holda ham murojaat etish mumkin. Statistik komponentlarga nafaqat obyekt ismi, balki sinf ismi orqali ham murojaat etish mumkin.

<sinf_nomi> : : <komponenta_nomi>

Masalan

complex:: count=5;

Lekin shunday murojaat faqatgina *public* komponentlarga tegishli.

private statik komponentlarga tashqaridan murojaat etishda funksiya – statik komponentlardan foydalaniladi.

Bu funksiyalarni sinf ismi orqali chaqirish mumkin.

<sinf_nomi> : : <statik_funksiya_nomi>

Misol.

```
#include <iostream.h>
class TPoint
{
```

```
  double x,y;
  static int N;  // statik maydon: nuqtalar soni
   public:
  TPoint(double x1 = 0.0,double y1 = 0.0){N++; x = x1; y = y1;}
  static int& count(){return N;} // statik komponenta-funksiya
};
int TPoint : : N = 0; //statik maydon initsializatsiyasi
void main(void)
{TPoint A(1.0,2.0);
  TPoint B(4.0,5.0);
  TPoint C(7.0,8.0);
  cout<< \nAniqlangan"<<TPoint : : count()<<"nuqtalar"; }
```

Obyektlar massivi ta'riflash uchun sinf ko'zda tutilgan (parametrsiz) konstruktorga ega bo'lishi kerak.

Obyektlar massivi ko'zda tutilgan konstruktor tomonidan, yoki har bir element uchun konstruktor chaqirish yo'li bilan initsializatsiya qilinishi mumkin.

```
class complex a[20]; //ko'zda tutilgan parametrsiz konstruktorni chaqirish
class complex  b[2]={complex (10),complex (100)};//oshkor chaqirish
```

Nazorat uchun savollar

1. C++tilida class, struct i union orasida qanday farq bor?
2. Sinf usullarini qo'shimcha yuklash mumkinmi?
3. Konstruktorlar va destruktorlar vazifasini ko'rsating.
4. Obyektlar massivi yaratilganda qanday konstruktorlar chaqiriladi?
5. Destruktorlar qanday chaqiriladi?
6. Statik komponentalar xususiy bo'lishi mumkinmi?

6 bob. SINFLAR VA KO‘RSATKICHLAR

6.1. 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. Masalan,

```
Type
   TchildClass=Class
        Fore: Integer;
        Ftwo: String;
        Fthree: Tobject;
End;
```

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. Masalan,

```
Var
 MyObject: TchildClass;
  Begin
```

```
MyObject.Fone:=16;
MyObject.Ftwo:=’qator qiymati’;
 End;
```

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.

6.2. Obyektlarga ko‘rsatkichlar

Obyektlar aniqlangandan so‘ng shu obyektlarga ko‘rsatkichlar belgilash mumkin. Masalan:

```
complex A(5.2,2.7);
complex* PA=&A;
```

Obyektning umumiy elementlariga murojaat uchun -> operatsiyani yoki ism almashtirish va nuqta operatsiyasidan foydalanish mumkin

```
*PA.real() yoki PA->real;
```

6.3. Sinf komponentlariga ko‘rsatkichlar

Sinf elementlariga ko‘rsatkichlar yordamida murojat etish mumkin. Buning uchun

.* i ->* amallari aniqlangan. Sinf maydoniga va usullariga ko‘rsatkichlar har xil tartibda aniqlanadi. Sinf usuliga ko‘rsatkich shakli:

turi (sinf_ismi::*ko‘rsatkich_ismi)(parametrlari);

Masalan, monstr sinfiga ko‘rsatkichning yozilishi qo‘yidagicha

```
int get_health() {return health;}
int get_ammo( ) {return ammo;}
//(xuddi shuningdek):
int (monstr:: *pget)();
```

Bunday ko‘rsatkichni funksiya parametri sifatida berish mumkin Bu esa funksiyaga usul ismini berish imkonini beradi:

```
void fun(int (monstr:: *pget)()){
```

```
(*this.*pget)();//  .* amali orqali funksiyani chaqirish
(this->*pget)();// ->* amali orqali funksiyani chaqirish
}
```

Koʻrsatkichni adresni olish orqali aniq bir usulga biriktirish ham mumkin:

```
//Koʻrsatkich qiymatini oʻzlashtirish:
pget=&monstr::get_health;
monstr Vasia.*p;
p = new monstr;
// .* amali orqali funksiyani chaqirish:
int Vasin.health - (Vasia.*pget)();
// ->* amali orqali funksiyani chaqirish:
int p_heaUh = (p->*pget)();
```

Qoʻyida sinflarning usullariga koʻrsatkichlardan foydalanish qoidalari koʻrsatilgan:

- Usulga koʻrsatkich, faqat mos sarlavhaga ega boʻlgan usullar adresi oʻzlashtirilishi mumkin.
- Sinfning statik usuliga koʻrsatkichni aniqlash mumkin emas.
- Oddiy funksiyaga koʻrsatkichni usuulga koʻrsatkichga almashtirish mumkin emas.

Oddiy funksiyaga koʻrsatkich kabi usullarga koʻrsatkich ismi noma’lum usullar hosil boʻlgandagina uni chaqirish uchun qoʻllaniladi. Usulga koʻrsatkich, oʻzgaruvchiga koʻrsatkich yoki oddiy funksiyaga koʻrsatkichdan farqi, usulga koʻrsatkich xotiraning aniq adresiga murojaat etmaydi.

U massivning indeksiga oʻxshab ketadi. Xotiraning aniq adresi usulga koʻrsatkich bilan ma’lum obyektga koʻrsatkichlar orqali amalga oshiriladi.

Sinf maydoniga koʻrsatkich shakli:

ma’lumotlar_turi(sinf_ismi::*koʻrsatkich_ismi);

Koʻrsatkichni aniqlashda uni initsializatsiyalashtirish shaklida ham berish mumkin:

&sinf_ismi::maydon_ismi;// maydon public boʻlishi kerak

Agar health maydoni public kabi aniqlanganda, uning ko‘rsatkichi quyidagi shaklda bo‘lar edi:

```
int (monstr::*phealth)=&monstr::health:
cout « Vasia.*phealth; //.* amali orqali murojaat etish
cout « p->*phealth; // ->* amali orqali murojaat etish
```

Bu yerda shunga e’tibor etish kerakki sinflarning maydoniga ko‘rsatkichlar oddiy ko‘rsatkichlar bo‘lmaydi, chunki qiymat o‘zlashtirilayotganda ular xotiraning ma’lum adresiga murojaat etmaydi.

Xotira sinflar uchun emas balki sinf obektlari uchun ajratiladi.

6.4. 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. X sinfni ekvivalent ko‘rinishda shunday tavsiflash mumkin:

```
class x {
int m;
public:
int readm() { return this->m; }
};
```

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. Quyida this ko‘rsatkich uchun misol keltiramiz:

```
monstr & the_best(monstr &M){
if( health >M.health) return nhis;
return M;
... monstr Vas1a(50). Super(200);
//Best yangi obyekti Super maydoni qiymatlari bilan initsializiya qilinmoqda:
monstr Best = Vasia.the_best(Super);
```

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:

```
void curednt health. 1nt ammo){
this -> health += health; // this dan foydalanish
monstr:: ammo += ammo; // :: amalidan foydalanish
```

6.5. Self ko‘rsatkichi

Sinfga birlashtirilgan protsedura va funksiyalarga usullar deyiladi. Masalan:

```
Type
   TchildClass=Class
        Fore: Integer;
        Ftwo: String;
        Fthree: Tobject;
        Function FirstFunc(x:Real):Real;
        Procedure SecondProc;
End;
```

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. Masalan:

// TPerson sinfi Show usuli

```
procedure TPerson.SHow;
begin
Write ( 'Nom:' + fname + #13+ 'Adres:' + faddress ); end;
```

Usul tanasida obyekt maydonlariga murojaat kilinganda obyekt nomi ko‘rsatilmaydi. Usulga murojaat qilish dasturda uning nomini ko‘rsatish bilan bajariladi. Masalan:

```
Var
  MyObject: TchildClass;
  y: Real;
   Begin
 . . . . . .
  MyObject. SecondProc;
  y:=MyObject.FirstFunc(3.14);
   End;
```

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.

```
procedure TPerson.Tproc(Fore:Integer);
begin
self.Fore:=Fore;
end;
```

Nazorat uchun savollar

1.Sinf bu nima?

2.Sinf va obyekt orasida qanday farq bor?

3. Inkapsulyatsiya nima uchun kerak?

4. Obyekt maydonlariga murojaat qanday amalga oshiriladi?

5.Xizmatchi so‘z private nima uchun ishlatiladi?

7 bob. SINFLAR ORASIDA MUNOSABATLAR

7.1. 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.

Misol:

```
class A
{  int i;
public:
  A(int ii) {i=ii;}
. . .
};
```

A sinfni o‘z ichiga kamrab olgan sinf:

```
class B
{  int k;
A member;
public:
  B(int ii, int kk): member(ii){k=kk;}
. . .
};
```

Boshqa a’zolar (agarda ular mavjud bo‘lsa) konstruktorlari uchun shuning singari parametrlarni belgilash mumkin:

```
class classdef {
  table members;
  table friends;
  int no_of_members;
  // ...
  classdef(int size);
  ~classdef();
};
```

A’zolar uchun parametrlar ro‘yxati bir biridan vergul (ikkita nuqta emas) yordamida ajratiladi, ammo a’zolar uchun initsializatorlar ro‘yxatini ixtiyoriy tartibda belgilash mumkin:

```
classdef::classdef(int size)
: friends(size), members(size), no_of_members(size)
```

```
{
  // ...
}
```

Konstruktorlar sinf tavsifida belgilangan tartibda chaqiriladi.

7.2 Sinflar do‘stlari

C++da aniq sinf do‘stlariga bu sinfning xususiy elementlariga murojaat etish imkonini beradi. C++ da bitta sinf yoki funksiya ikkinchi sinfga do‘stona sinfligini ko‘rsatish uchun friend kalit so‘zidan foydalanish va do‘stona sinf ismini ikkinchi sinf tavsifiga kiritish lozim.

Do‘stona funksiya va sinflarni ta’riflash:

friend <funksiya prototipi >

friend <sinf nomi >

Masalan, quyidagi book sinfi librarian sinfini o‘ziga do‘stona sinf deb belgilagan:

```
class book
      { char title [64] ;
char author[64];
char catalog[64];
public:
book (char *, char *, char *);
void show_book(void);
friend librarian;
};
```

Shuning uchun librarian sinf obyektlari book sinfning xususiy elementlariga, nuqta operatoridan foydalangan holda, to‘g‘ridan to‘g‘ri murojaat etishi mumkin:

```
class librarian
{
public:
  void change_catalog(book *, char *);
```

```
  char *get_catalog(book);
};
      void librarian::change_catalog(book *this_book, char *new_catalog)
      {
  strcpy(this_book->catalog, new_catalog);
}
      char *librarian: :get__catalog(book this_book)
      {  static char catalog[64];
          strcpy(catalog, this_book.catalog);
          return(catalog) ;
        }
```

Ko‘rib turganimizdek dastur librarian klassining change_catalog funksiyasiga book obyektini adres orqali bermoqda. Bu funksiya klassning book elementini o‘zgartirgani uchun, dastur parametrni adres orqali uzatishi va undan so‘ng ushbu klass elementiga murojat uchun ko‘rsatkich ishlatmog‘i lozim. Book klassi aniqlanishidan friend operatori o‘chirib yuborilsa C++ kompilyatori har gal book klassi xususiy ma’lumotlariga murojatda sintaksik xato xaqida xabar chiqaradi.

7.3. 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:

```
      class book
      {
public:
  book(char *, char *, char *);
  void show_book(void);
```

```
  friend char *librarian::get_catalog(book);
  friend void librarian: :change_catalog( book *, char *);
private:
  char title[64];
  char author[ 64 ];
  char catalog[64];
};
```

Ko‘rib turganimizdek friend operatorlari xususiy elementlarga murojat qiluvchi hamma do‘st funksiyalarini to‘liq prototiplarini o‘z ichiga oladi.

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:

```
    class librarian
    {
public:
  void change_catalog(book *, char *);
  char *get_catalog(book);
};
```

book klassi aniqlanishini librarian klassi aniqlanishidan oldini qo‘yib bo‘lmagani uchun. C++ book klassini e’lon qilish imkonini beradi va shu bilan u kompilyatorga bunday klass borligi xaqida xabar beradi va keyinroq o‘zi ham aniqlanadi. Quyida buni qanday amalga oshirish keltirilgan:

```
    class book; // sinf elon qilinishi
```

7.4. 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.

Nazorat uchun savollar

1. Qachon A va V sinflari «mijoz – yetkazuvchi » munosabatida bo‘ladi?
2. Qachon A va V sinflari «ota - o‘g‘il» munosabatlarida bo‘ladi?
3. "Egalik" va "bir xillik" munosabatlari qaysi munosabatlar tipiga mansub?
4. Do‘stona funksiya va sinflar aniqlanish shaklini ko‘rsating.
5. Sinf do‘stlaridan nima uchun foydalaniladi?
6. Do‘stona sinfni e’lon qilish sinfning qaysi seksiyasida ekanligi axamiyatlimi?

8 bob. SINFLARDA VORISLIK

8.1. Sinflarda vorislik. Sodda 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.

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 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.

8.2. 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.

Agarda V sinf quyidagicha aniqlangan bo‘lsa:

```
class B { protected: int t;
public: char u;
};
```

unda quyidagi xosila sinflarni kiritish mumkin:

```
class M: protected B { ... }; // t, va u protected sifatida merosxo‘r
class P: public B { ... }; // protected, va u- public sifatida merosxo‘r
class D: private B { ... }; // t, va u  private sifatida merosxo‘r
struct F: private B { ... }; // t, i u private sifatida merosxo‘r
struct G: public B { ... }; t - protected, va u – public sifatida merosxo‘r
```

8.3. Vorislikda konstruktorlar va destruktorlar

Konstruktorlar meros bo‘lmagani uchun, xosila sinfni yaratishda undan meros bo‘lgan ma’lumot – a’zolari asosiy(bazaviy) sinf konstruktori orqali initsializatsiyalanishi lozim. Asosiy sinf konstruktori avtomatik ravishda chaqiriladi va xosila sinfni konstruktoridan oldin bajariladi. Asosiy (bazaviy) sinfni konstruktorining parametrlari xosila sinfni konstruktorni aniqlashda ko‘rsatiladi.

Shunday qilib argumentlarni xosila sinfni konstruktoridan asosiy (bazaviy) sinfni konstruktoriga uzatish vazifasi bajariladi.

Masalan.

```
class Basis
{ int a,b;
public:
Basis(int x,int y){a=x;b=y;}
};
class Inherit:public Basis
{int sum;
public:
Inherit(int x,int y, int s):Basis(x,y){sum=s;}
};
```

Sinf obyektlari pasdan tepaga qarab konstruktorlanadi: avvalo asosiy(bazaviy), keyin esa kopmonent – obyektlar (agarda ular mavjud bo‘lsa), undan keyin esa xosila sinfning o‘zi. Shunday qilib, xosila sinfning obyekti quyi obyekt sifatida asosiy (bazaviy) sinf obyektini o‘z ichiga oladi. Obyektlar teskari tartibda o‘chiriladi: avvalo xosila, keyin uning komponent – obyektlari, undan keyin esa asosiy(bazaviy) obyekt.Shunday qilib obyektni o‘chirish tartibi uning konstruktorlash tartibiga nisbatan teskari bo‘ladi.

8.4. 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. Misol:

```
class base
{
public:
  virtual void print(){cout<<“\nbase”;}
. . .
};
class dir : public base
{
public:
  void print(){cout<<“\ndir”;}
};
void main()
{
  base B,*bp = &B;
  dir D,*dp = &D;
  base *p = &D;
  bp –>print(); // base
  dp –>print(); // dir
  p –>print(); // dir
}
```

Bu misolda avlod sinfi obyektiga adres qiymat sifatida berilgan ajdod sinf turidagi ko‘rsatkich yoki ilova orqali avlodda qo‘shimcha yuklangan usulni chaqirishga e’tibor berish lozim. Agar funksiya novirtual bo‘lsa ajdod sinf usuli, virtual bo‘lsa avlod sinf usuli chaqiriladi.

SHunday qilib ajdod sinf turidagi ko‘rsatkich orqali virtual usul chaqirish natijasi shu ko‘rsatkich qiymati ya’ni chaqiriq bajarilayotgan obyekt turi bilan aniqlanadi.

Qaysi virtual funksiyani chaqirish ko‘rsatkich turiga emas shu ko‘rsatkich qaratilgan(dastur bajarilish jarayonida) obyekt turiga bog‘liq.

Hech bo‘lmasa bitta sof (bo‘sh) virtual funksiyaga ega bo‘lgan sinf abstrakt sinf deyiladi. Quyidagi tavsifga ega bo‘lgan komponentali funksiya sof virtual funksiya deyiladi:

virtual <tip> <funksiya_nomi>(<formal_parametrlar_ro‘yxati>) = 0;

Sof virtual funksiya xech narsa qilmaydi va uni chaqirib bo‘lmaydi. Uning qo‘llanilishi – xosila sinflarda uning o‘rnini egallovchi funksiyalar uchun asos bo‘lish. Abstrakt sinf esa xosila sinf uchun asosiy (bazaviy) sinf sifatida ishlatilishi mumkin.

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:

```
class shape {
  // ...
public:
  virtual void rotate(int) = 0; // sof virtual funksiya
```

```
  virtual void draw() = 0;      // sof virtual funksiya
};
```

Abstrakt sinfni faqat boshqa sinf ajdodi sifatida ishlatish mumkin:

```
class circle : public shape {    int radius;
public:
  void rotate(int) { }
 // qayta ta'riflash shape::rotate
  void draw();
 // qayta ta'riflash shape::draw
  circle(point p, int r);
};
```

Agar sof virtual funksiya xosila sinfda to‘liq ta’riflanmasa, u xosila sinfda xm sof virtual bo‘lib qoladi, natijada xosila sinf ham abstrakt sinf bo‘ladi.

Abstrakt sinflar realizatsiya detallarini aniqlashtirmasdan faqat interfeysni ko‘rsatish uchun ishlatiladi. Masalan operatsion tizimda qurilma drayveri abstrakt sinf sifatida berilishi mumkin:

```
class character_device {
public:
   virtual int open() = 0;    virtual int close(const char*) = 0;
   virtual int read(const char*, int) =0;    virtual int write(const char*, int) = 0;
   virtual int ioctl(int ...) = 0;
   // ...
};
```

Drayverlar character_device sinfining ajdodlari sifatida kiritilishi mumkin.

8.5. Polimorfizm. Ko‘plik vorislik. Interfeyslar

Bu sinf ketma-ket(to‘g‘ri-to‘g‘ri) baza sinfi, agar u sinflarni aniqlashda baza ro‘yxatidan chiqadi.Agar ba’zi hollarda A sinf V sinfning bazasini ifodalasa va S uchun V baza bor, unda V sinf S uchun to‘g‘ridan-to‘g‘ri bazasi hisoblanadi, A sinf esa S sinf uchun to‘g‘ri bo‘lmagan baza bo‘lib g‘isoblanadi. xa komponentiga

murojat qilganda A sinfga kiruvchi va V va S sinflarga izchil tarzida voris bo‘ladi,S sinfga A::xa tarzida, V::xa tarzida e’lon qilish mumkin. Ikkala konstruksiya A sinfning elementi xa ga murojat qilishni ta’minlaydi. Pastda sinflarni tasvirlashda qabul qilingan bazalar ishlab chiqilgan. Xuddi shu tartibda ularni kompilyator e’lon qilishini ko‘ramiz va ularni teksti dasturda joylashadi.

Sinflar bir nechta ketma-ket sinflardan tashkil topishi mumkin, sinf bazasida ixtiyoriy son yo‘qolishi mumkin, misol uchun,

```
class X1 { ... };
class X2 { ... };
class X3 { ... };
class Y1: public X1, public X2, public X3 { ... };
```

Bir necha to‘g‘ri baza sinflari mavjud bo‘lib, ular ko‘plik vorislari deb nomlanadi.

Ko‘plik vorislarida ketma-ket bazada hech qanday sinf bittadan ortiq ishlatilishi mumkin emas. Bitta sinf to‘g‘ri bo‘lmagan sinfda bir necha marta ishlatilishi mumkin:

```
class X { ...; f () ;  ... };
class Y: public X { ... };
class Z: public X { ... };
    class D: public Y, public Z { ... };
```

Bu misolda X sinf D sinfning ikki marta o‘rtacha vorisi bo‘ladi. Bizning misolimizda ikkita qiymati qatnashadi, va shuning uchun bir qiymatli bo‘lmagan D sinfning obyekti X sinfning aniq komponentiga murojat qulayligini bartaraf qilish kerak, uni to‘liq kvalifikatsiyasi D::Y::X::f() yoki D::Z::X::f(). D sinfning obyekti ichida sodda ko‘rinish Y::X::f() yoki Z::X::f(), lekin bu ham kvalifikatsiyani mazmunidir.

Bir xil nOYDagi obyektlarni bartaraf qilishda to‘g‘ri bo‘lmagan sinf bazalari ko‘plik vorislari, bu sinf bazalari virtual deb e’lon qilinadi. Buning uchun sinf bazalari ro‘yxatida oldingi sinf nomini virtual kalit so‘zini ishlatish kerak. Misol uchun X sinfi virtual baza sinfi bo‘la oladi bunaqa ko‘rinishda yozilganda:

```
class X { ... f() ; ... };
class Y: virtual public X { ... };
class Z: virtual public X { ... };
    class D: public Y,  public Z { ... };
```

Endi D sinf X dan faqat bitta nusxada bo‘ladi, ruxsat etilgan to‘g‘ri tenglikdan Y va Z sinflarini ifodalaydi.

Nazorat uchun savollar

1. Nima uchun avval ajdod sinf konstruktorlari chaqirilib, so‘ngra avlod sinf konstruktori chaqiriladi?
2. Nima uchun destruktorlar konstruktorlarga nisbatan teskari tartibda chaqiriladi?
3. Vorislikda ajdod sinf spetsifikatori sifatida protected ko‘rsatilishi mumkinmi?
4. Sinflar bibliotekasini qurishda vorislikdan qanday foydalaniladi?
5. Xususiy deb e’lon qilingan komponentalarga boshqa sinf usullari orqali murojaat qilish mumkinmi?

9 bob. STANDART AMALLARNI QO‘SHIMCHA YUKLASH

9.1. Qo‘shimcha yuklash ta’rifi

Standart amallarni (masalan +) qo‘shimcha yuklash biror sinf bilan birga qo‘llashda mazmunini o‘zgartirishdan iboratdir.

Standart amallarni qo‘shimcha yuklash maxsus funksiya – komponenta kiritish yo‘li bilan amalga oshiriladi. Qo‘shimcha yuklash til standartiga asosan amalga oshiriladi, amallar belgisi va operandlar soni o‘zgarmaydi.

Amallarni qo‘shimcha yuklash uchun quyidagi ta’rifdan foydalaniladi:

<operator amal> (<operandlar ro‘yxati>)

quyidagi amallarni qo‘shimcha yuklash mumkin:

+ - * / % ^ & | ~ !

= < > += -= *= /= %= ^= &=

|= << >> >>= <<= == != <= >= &&

|| ++ -- [] () new delete

Bu amallar ustivorligi va ifodalar sintaksisini o‘zgartirish mumkin emas. Masalan unar amal % yoki binar ! amalni kiritish mumkin emas. Funksiya amal har qanday funksiya kabi ta’riflanadi va chaqiriladi.

Standart tiplar uchun to‘rt amal ("+", "-", "*" i "&") ham unar ham binar amal sifatida ishlatiladi va qo‘shimcha yuklanadi.

Xamma qo‘shimcha yuklangan amallar uchun operator() amalidan tashqari, ko‘zda tutilgan argumentlardan foydalanish mumkin emas.

Amallar xossalaridan ba’zilaridan foydalaniladi. Xususan operator=, operator [ ], operator ( ) va operator-> nostatik komponenta – funksiya bo‘lishi lozim.

Operator - funksiya yoki sinf komponentasi bo‘lishi kerak yoki juda bo‘lmasa bitta parametri sinf obyekti bo‘lishi kerak (new va delete amallarini qo‘shimcha yuklovchi funksiyalar uchun bu shart emas).

Operator - funksiya, birinchi parametri asosiy turga tegishli bo‘lsa, funksiya-komponenta bo‘lolmaydi.

C++tilida quyidagi amallarni qo‘shimcha yuklash mumkin emas:

. sinf obyekti a’zosiga murojaat;

.* ko‘rsatkich orqali murojaat;

?: shartli amal;

::ko‘rinish soxasini ko‘rsatuvchi amal;

sizeof hajmni hisoblash amali;

preprotsessor amali.

9.2. Binar amallarni qo‘shimcha yuklash

Ixtiyoriy ⊕ binar amali ikkita usulda tavsiflanish mumkin: yoki bitta parametrli funksiya komponentasi sifatida, yoki ikkita parametrli global (do‘stona bo‘lishi mumkin) funksiya sifatida.

Birinchi xolatda x⊕y ifoda x.operator⊕(y), ikkinchi holda esa operator ⊕(x,y) chaqirilishini bildiradi.

Aniq sinf doirasida qayta yuklangan operatsiyalar faqat parametrli nostatik komponentli funksiya orqali qayta yuklanadi. Sinfdagi chaqiriladigan obyekt avtomatik ravishda birinchi operand sifatida qabul qilinadi.

Misol:

```
class complex
{
  double re, im;
 public:
  complex(double r, double i) { re=r; im=i; }
  complex operator+(complex);
  complex operator+(double);
 complex& operator+=(complex);
 complex& operator+=(double x);
};
inline complex operator+(complex b) //complex+complex
{ complex c; c.re=re+b.re; c.im=im+b.im; return c;}
```

```
inline complex operator+( double x) //complex+double
{ return complex(re+x, im);}
inline complex& complex::operator+=(complex b) //complex+=complex
 {  complex c; re += b.re; im += b.im; c.re=re; c.im=im; return c; }
inline complex& complex::operator+=(double x) //complex+=double
{  re += x; return *this; }
```

Funksiya – amalni chaqirish:

```
  complex c = a + b;     // qisqa yozuv
  complex d = a.operator+(b); // oshkora chaqirish
```

Shuni ta’kidlab o‘tish kerakki double+complex tipidagi qo‘shishni bu usulda qo‘shimcha yuklash mumkin emas.

Sinfdan tashqari yuklanadigan operatsiyalar ikkita operandga ega bo‘lishi kerak. Ulardan biri sinf tipiga ega bo‘lishi lozim.

Qiymat qaytarish uchun vaqtinchalik o‘zgaruvchidan foydalaniladi. Bu maqsadda konstruktordan:

```
inline complex operator +(complex b)
{ return complex(re+b.re, im+b.im)}
```

va this ko‘rsatkichidan foydalanish mumkin:

```
inline complex operator+=(complex b)
{ re+=b.re, im+=b.im; return *this}
```

Misol:

```
class complex
{
  double re, im;
 public:
  complex(double r, double i) { re=r; im=i; }
  friend  complex operator+( complex , complex);
  friend  complex operator+( complex ,double);
  friend  complex operator+( double, complex);
```

```
};
inline complex operator+(complex a, complex b) // complex+complex
{ complex c; c.re=a.re+b.re; c.im=a.im+b.im; return c;}
inline complex operator+(complex a, double x) //complex+double
{ complex c; c.re=a.re+x; c.im=im; return c;}
inline complex operator+(double x, complex a) // double + complex
{ complex c; c.re=a.re+x; c.im=im; return c;}
```

Funksiya – amalni chaqirish:

```
  complex c = a + b;     // qisqa yozuv
  complex d = operator+(a,b); // oshkora chaqirish
```

9.3. Unar amallarni qo‘shimcha yuklash

Ixtiyoriy ⊕ unar amali ikkita usulda tavsiflanish mumkin: yoki paramatrsizli funksiya komponentasi sifatida, yoki bitta parametrli global (do‘stona bo‘lishi mumkin) funksiya sifatida. Birinchi xolatda ⊕ Z ifoda Z.operator ⊕ (), ikkinchi xolatda esa operator ⊕(Z) chaqirilishini bildiradi.

Aniq sinf doirasida qayta yuklangan unar operatsiyalar faqat parametrsizli nostatik komponentli funksiya orqali qayta yuklanadi. Sinfdagi chaqiriladigan obyekt avtomatik ravishda operand sifatida qabul qilinadi.

Sinf doirasidan tashqarida qayta yuklangan unar operatsiyalar (global funksiya kabi) sinf tipdagi bitta parametrga ega bo‘lishi lozim. Shu parametr orqali uzatiladigan obyekt operand sifatida qabul qilinadi.

Sintaksis:

a) birinchi holda (sinf doirasida tavsiflash):

<qaytariluvchi_qiymat_tipi> operator <amal_belgisi>

b) ikkinchi holda (sinf doirasidan tashqari tavsiflash):

<qaytariluvchi_qiymat_tipi> operator <amal_belgisi>

(<tip_identifikatori>)

9.4. Inkrement va dekrement amallarini qo‘shimcha yuklash

C ++ tilining zamonaviy versiyalarida prefiks ++ va -- operatsiyalarni qo‘shimcha yuklash boshqa operatsiyalarni yuklashdan fark qilmaydi, Postfiks shakldagi ++va -- amallarini kayta yuklaganda yana bir int tipidagi parametr kiritilishi kerak.Agar qo‘shimcha yuklash uchun global funksiya ishlatilsa uning birinchi parametri sinf tipiga, ikkkinchi parametri int tipiga ega bo‘lishi kerak.

Dasturda postfiks ifoda ishlatilganda butun parametr ham qiymatga ega bo‘ladi.

Quyida postfiks va prefiks ++ va – amallari uchun funksiya – amallarga misollar keltirilgan.

Misol:

```
complex &complex::operator++() // prefiks uchun komponenta
{ re++; return *this;}
complex &complex::operator--(int k) // postfiks uchun komponenta
{ re--; return *this;}
complex &operator++(complex& a) // prefiks uchun do‘stona
{ a.re++; return a;}
complex &operator++(complex& a, int k) // postfiks uchun do‘stona
{ a.re++; return a;}
```

9.5. Indekslash va funksiyani chaqirish amallarini qo‘shimcha yuklash

Dumaloq qavs () amalini qo‘shimcha yuklash sinf obyektiga funksiya chaqirig‘i sintaksisini qo‘llashga imkon beradi. Operandlar soni ixtiyoriy bo‘lishi mumkin. Kvadrat qavs massiv elementi sintaksisini qo‘llashga imkon beradi.

```
//------ Simvolni ajratish amali -------------------
char  string::operator()(int n) { if n<strlen(Str) return Str[i];
else return ‘\0’;}
  //------ Ostki satrni ajratish amali -------------------
  string  string::operator()(int n1, int n2) {
  string  tmp = *this;
```

```
delete  tmp.Str;
tmp.Str = new char[n2-n1+1];
strncpy(tmp.Str, Str+n1, n2-n1);
return tmp;
}
```

9.6. Qiymat berish va initsializatsiya

Qiymat berish va initsializatsiya turli amallardir. Ayniqsa destruktor aniqlanganda bu muximdir. Biror X tipidagi obyektni insializatsiya qilish nusxa olish konstruktori yordamida amalga oshiriladi. Satr – bu simvollar vektoriga ko‘rsatkich.

Vektor konstruktor tomonidan yaratilib, destruktor bilan o‘chirilganda muammo tug‘ilishi mumkin:

```
string s1(10);
string s2(20)
s1 = s2;
```

Bu yerda ikki simvolli vektor joylashadi, lekin s1 = s2 qiymat berish natijasida biri o‘chirilib, ikkinchisi nusxasi bilan almashtiriladi. Funksiyadan chiqishda s1 va s2 uchun destruktor chaqiriladi va bitta vektor ikki marta o‘chiriladi. Bu muammoni xal qilish uchun qiymat berish amalini qo‘shimcha yuklash lozim:

```
string& string::operator=(const string& a)
{   if (this !=&a) {   // opasno, kogda s=s
      delete p;
      p = new char[size=a.size];
      strcpy(p,a.p);
   }
   return *this; }
```

Foydalanuvchi qiymat berish operatori initsializatsiya qilinmagan obyektga qo‘llanmaydi. Bu holda p ko‘rsatkich tasodifiy qiymatga ega bo‘ladi.

Initsializatsiya masalasini xal qilish uchun nusxa olish konstruktori kiritish lozim:

```
string::string(const string& a)
{   p=new char[size=sz];
    strcpy(p,a.p); }
```

Nazorat savollari:

1. Postfiks va prefiks amallar orasida qanday farq bor?
2. Qo‘shimcha yuklangan amallar qanday ikki usulda aniqlanadi?
3. Global do‘stona funksiya yordamida xamma amallarni qo‘shimcha yuklash mumkinmi?
4. Qaysi xolatda amalni global funksiya yordamida qo‘shimcha yuklash mumkin?
5. Funksiya operator da “sinf” yoki “sinfga ilova” tipidagi parametr ishlatish shartmi?
6. Unar va binar amal-funksiyalar sintaksisi farqi nimadan iborat?

10 bob. FUNKSIYALAR VA SINFLAR SHABLONLARI

10.1. Funksiyalar shablonlari

Funksiya shablonlari (parametrlangan turlar) bog‘langan funksiyalar oilasini tuzish imkonini beradi. Shablon kiritilishi uchun, hosil qilingan funksiya avtomatlashtirish, har xil tipli ma’lumotlarni qayta ishlashdan iborat. Masalan, algoritm tartiblash uchun har qaysi funksiyani o‘zini aniqlovchi tipi qo‘llaniladi. Funksiya shabloni bir marta aniqlanadi, lekin parametrli aniqlashda va hokazo, ma’lumotlar tipi shablon parametrlari orqali beriladi. Shablon formati:

template <class tip_nomi [,class tip_nomi]>

<funksiya_sarlavxasi>

{ <funksiya tanasi>}

Shablon funksiyasining asosiy parametrlarining xossasi.

Funksiyalar shablonlari parametrlarining asosiy xususiyatlari:

1. Parametrlar nomlari shablonning butun ta’rifi bo‘ylab unikal bo‘lmog‘i lozim.

2. Shablon parametrlarining ro‘yxati bo‘sh bo‘la olmaydi.

3. Shablon parametrlari ro‘yxatida har biri class so‘zidan boshlanadigan bir nechta parametr bo‘lishi mumkin.

Misol:

template <class T>T max(Tx, Ty){return(x>y)? x:y;}:

bunda<class T> shablonining argumenti tomonidan taqdim etilgan ma’lumotlar turi har qanday bo‘lishi mumkin. Undan dasturda foydalanishda kompilyator tax funksiyasi kodini bu funksiyaga uzatilayotgan parametrlarning faktik turiga muvofiq generatsiya qiladi:

```
int i;
Myclass a,b;
int i=max(i, 0);//argumentlar turi int
myclass m=max(a, b);// argumentlar turi myclass
```

Faktik turlar kompilyatsiya paytida ma’lum bo‘lishlari kerak. Shablonlarsiz max funksiyasini ko‘p martalab ortiqcha yuklashga to‘g‘ri kelar edi, ya’ni, garchi barcha funksiya versiyalarining kodlari bir xil bo‘lsa ham, har bir qo‘llanayotgan tur uchun aloxida ortiqcha yuklash kerak bo‘lar edi.

10.2. Sinflar shablonlari. Sinf shablonlarning asosiy xossalari

Sinf shablonlari (o‘zgacha parametrlangan sinf) avlodga oid sinfni tuzish uchun ishlatiladi. Tuzish qoidalarni va ayrim obyektlarni formatini aniqlovchi sinf kabi, sinf shablonlari ayrim sinflarni tuzish usullarini aniqlaydi. Shablondagi sinf ta’rifida sinf ismi ayrim sinflarning emas oilaviy sinflarning parametrlangan ismi bo‘ladi.

Parametrlangan sinfning e’lon etish umumiy shakli:

template <class tip_dannых> class imya_klassa { . . . };

Parametrlangan sinf funksiyalarining komponentalari avtomatik ravishda parametrlangan bo‘ladi. Ularni template yordamida parametrlangan sifatida e’lon etish shart emas.

Parametrlangan sinfda tavsiflangan do‘stona funksiyalar avtomatik ravishda parametrlangan funksiyalar bo‘lmaydi, ya’ni ko‘rsatilmagan holda shunday funksiyalar berilgan shablon bo‘yicha tashkil etilgan barcha sinflar uchun do‘stona bo‘ladi.

Agarda friend-funksiya o‘z tavsifida parametrlangan sinf tipdagi parametrga ega bo‘lsa, unda berilgan shablon bo‘yicha yaratilgan barcha sinflar uchun xususiy friend–funksiyasi mavjud.

Parametrlangan sinf doirasida friend-shablonlarni (do‘stona parametrlangan sinflar) tavsiflash mumkin emas.

Bir tarafdan, shablonlar shablonlardan xosil (meros) bo‘lgandek, oddiy sinflardan ham xosil (meros) bo‘lishi mumkin. Ikkinchi tarafdan esa ulardan boshqa shablonlar va sinflar uchun bazoviy sifatida foydalanishi mumkin.

Sinf a’zosi bo‘lgan funksiyalar shablonlarini virtual sifatida tavsiflash mumkin emas.

Lokal sinflar o‘z elementlar sifatida shablonlarni o‘z ichiga olish mumkin emas.

10.3. Parametrlangan sinflarning komponent funksiyalari

Sinf shablonining tavsifidan tashqarida joylashgan sinf shablonining komponentli funksiyasini amalda oshirishda quyidagi ikkita elementni qo‘shimcha kiritish lozim:

– Tavsiflash template kalitli so‘zdan boshlanishi lozim, undan so‘ng burchakli qavislarda sinf shablonni tavsifida ko‘rsatilgan tiplar_parametrlarining_ruyxati keladi.

– Ko‘rish soxasi operatsiyasidan (::) oldinda bo‘lgan sinf ismidan so‘ng shablonning parametrlar_ismlari_ro‘yxati kelishi lozim.

template<tiplar_ro‘yxati><qaytariluvchi_qiymat_tipi> <sinf_nomi> <parametrlar_nomlari_ro‘yxati > : :

<funksiya_nomi>(<parametrlar_ro‘yxati>){ . . . }

Sinf obyektlari bilan ishlash uchun vector qo‘shimcha yuklangan shablon sinfi:

```
template<class T > class vector
{
T *data;
int size;
public:
array(int k) {size =k; data = new T[size]};
T&  operator[](int i){return data[i];}
int size() { return size; }
~array () { delete []data; }
void input_ vector ();
void show_ vector ();
};
template<class T > void vector <T >:: input_array()
{
```

```
for (int i = 0; i < index; i++) {cin>>data[i] ; cout << ' ';}
}
template<class T > void vector <T >:: show_array()
{
for (int i = 0; i < index; i++) cout << data[i] << ' ';
}
```

10.4. Funksiya uchun shablon turi

Shablon sinflarini qo‘llanilishi shablon funksiyasini a’zosini ifodalaydi. Parametrlari shablon sinflarini ifodalovchi global shablon funksiyalar - algoritmlarini aniqlash mumkin. Masalan oddiy shablonni pufaksimon algoritm orqali tartiblashni shunday aniqlash mumkin.

```
template<class T> void bubble_sort(Vector<T>& v)
{  unsigned n = v.size();
  for (int i=0; i<n-1; i++)
      for (int j=n-1; i<j; j--)
          if (v[j] < v[j-1])
{  // menyayem mestami v[j] i v[j-1]
             T temp = v[j];
             v[j] = v[j-1];
             v[j-1] = temp;
          }
    }
```

Vector tipini qismiga tenglik funksiyasini berish mumkin emas, balki unga ikkinchi parametr sort() funksiyasini berish kerak. Bu parametr obyekt sinfini ifodalaydi, qaysiki tenglik operatsiyasini qayta aniqlashda.

```
template<class T, Compare >
void bubble_sort(Vector<T>& v, Compare & cmp)
{
  unsigned n = v.size();
```

```
  for (int i = 0; i<n-1; i++)
      for ( int j = n-1; i<j; j--)
          if (cmp.lessthan(v[j],v[j-1])) {
             T temp = v[j];
             v[j] = v[j-1];
             v[j-1] = temp;
          }
  }
```

for_each() algoritm yordamida har xil ko‘rinishidagi qayta ishlashni va har bir elementni modifikatsiyasini ko‘rish mumkin.

```
  Template<class Item, class function>
  void for_each(vector<Item> a, function op)
  {
  for( int i=0; i<a.size(); i++ ) op(a[i]);
  }
```

for_each() algoritmining qo‘llanishi:

```
  # include <iostream.h>
  class StudentPrint
  {public:
  void operator() (Student elem){ if (elem.rating>5) elem.print();}
  };
  main()
  {
    vector<Student> coll(5);
     coll.input();
    StudentPrint cmp;
    for_each(coll, cmp);
    }
```

10.5. Yuqori darajali funksiyalar

Funksiya obyektlari – bu «kichik qavs» () operatsiya aniqlangan sinf nusxasi. Ba’zi bir xolatlarda funksiyani obyekt – funksiyalarga almashtirish qulaydir. Obyekt - funksiya funksiya sifatida ishlatilsa uni chaqirish uchun operator () foydalaniladi.

Misol:

```
class kub{
public:
double operator()(double x)
{return x*x*x;} };
```

Yuqori darajadagi algoritm bu shunday algoritmki bitta yoki bir nechta argumentlar funksional tipga tegishlidir.

Dixotomiya usuli yerdamida ixtiyoriy funksiya uchun [a,b] oralikda f(x)=0 tenglamani yechish masalasi misolida yuqori darajali funksiyani ko‘rib o‘tamiz. Bu maqsadda masalani yechadigan metodni tavsiflovchi sinf yaratamiz. Sinfni tavsiflovchi dastur kodini ko‘ramiz:

```
template <class T>
class FunctionZero
{
public:
static double dihotom(double a, double b, double eps, T f)
};
template <class T>
double FunctionZero<T>::dihotom(double a, double b, double eps, T f)
{
  float x, x1=a, x2=b;
  while (x2-x1)>eps
{x=(x1+x2)\2;
if (f(x)==0) return x;
if (f(x)>0) x1=x; else x2=x;
```

```
};
return x1;
}
```

Nazorat uchun savollari

1. Shablonlarlardan nima maqsadda foydalaniladi?
2. Funksiya shabloni asosiy xossalarini ko‘rsating.
3. Parametrlashtirilgan sinflar xossalarini ko‘rsating.
4. SHablon parametrlari ro‘yxati bo‘sh bo‘lishi mumkinmi?
5. Parametrlashtirilgan funksiya qanday chaqiriladi?
6. Parametrlashtirilgan sinflar xamma komponenta funksiyalari parametrlashganmi?
7. Sinf shabloni tashqarisida komponenta funksiyalar qanday aniqlanadi?

11 bob. OQIMLI SINFLAR

11.1. Oqimli sinflar iyerarxiyasi

C++da oqimli sinflar bibliotekasi ikkita bazaviy **ios** va **streambuf** sinflar asosida tuzilgan. streambuf sinfi kiritish-chiqarish fizik qurilmalari bilan xotirada joylashgan kiritish-chiqarish buferlarni o‘zaro bo‘g‘lanishini va tashkilini ta’minlaydi. Streambuf sinfining metodlarini va ma’lumotlarini dasturchi ochiq ishlatmaydi. Mavjud bo‘lgan sinflar asosida yangi sinflarni yaratishda dasturchiga ham sinfga murojaat etish ruxat etilgan.

ios sinfi formal kiritish chiqarish va xatolarni tekshirish vositalarga ega. Standart oqimlar (istream, ostream, iostream) terminal bilan ishlash uchun xizmat qiladi. Satrli oqimlar (istrstream, ostrstream, strstream) xotirada joylashtirilgan satrli buferlardan kiritish-chiqarish uchun xizmat qiladi. Faylli oqimlar(ifstream, ofstream, fstream) fayllar bilan ishlash uchun xizmat qiladi. Oqimli sinflar, ularning metodlari va ma’lumotlari dasturda murojaat etish ruxsatiga ega bo‘ladi, qachonki unga kerakli bosh fayl kiritilgan bo‘lsa.

iostream.h – ios, ostream, istream uchun.

strstream.h – strstream, istrstream, ostrstream uchun

fstream.h – fstream, ifstream, ofstream uchun

Quyidagi obyekt-oqimlar dasturda main funksiyasini chaqirish oldidan avvaldan aniqlangan va ochilgan bo‘ladi:

```
extern istream cin;    //Klaviaturadan kiritish standart oqimi
extern ostream cout; //Ekranga chiqarish standart oqimi
extern ostream cerr; //Xatolar xaqidagi xabar chiqarish standart oqimi
```

11.2. Oqimli sinflar usullari

Oqimdan qiritish uchun istream sinfdagi obyektlar ishlatiladi, oqimga chiqarish uchun - ostream sinfdagi obyektlar.

istream sinfda quyidagi funksiyalar tavsiflangan:

- istream get (char& S);

– istream dan S ga simvolni o‘qiydi. Xato xolatida S 0XFF qiymatini oladi.

– int get();

– istream dan keyingi simvolni chiqaradi. Faylni oxirini aniqlagach EOFni qaytaradi.

– istream& get(char* buffer,int size,char delimiter=’\n’);

Bu funksiya istreamdan simvollarni chiqaradi va ularni buferga nusxalaydi. Operatsiya yoki faylning oxiriga yetganda, yoki size fayllardan nusxa olgan jarayonda, yoki ko‘rsatilgan ajratuvchini aniqlaganda to‘xtaydi. Ajratuvchi esa nusxalanmaydi va streambuf qoladi. O‘qib bulingan simvollar ketma-ketligi hardoim nul simvol bilan tugatiladi.

– istream& getline(char* buffer,int size, char delimiter=’\n’);

– Ajratuvchi oqimdan chiqariladi, lekin, buferga kiritilmaydi. Bu esa satrlarni oqimdan chiqaruvchi asosiy funksiya. O‘kib chiqilgan simvollar nul simvoli bilan ta’momlanadi.

– istream& read(char* buffer,int size);

– Ajratuvchilarni qo‘llanmaydi va buferga o‘qilgan simvollar nul simvoli bilan tugamaydi.

– int peek();

– istream dan simvolni chiqarmasdan istreamga qaytaradi.

– int gcount();

Formatlanmagan oxirgi kiritish operatsiyasi vaqtida o‘qilgan simvollar sonini qaytaradi.

istream& putback(S)

Agar get doirasidagi streambuf obyektida bo‘sh fazo mavjud bo‘lsa, unda o‘sha yerga S simvoli joylashtiriladi.

istream& ignore(int count=1,int target=EOF);

Quyidagilar bajarilmaguncha istream dan simvol chiqarilaveradi:

– funksiya count simvollarni chiqarmaguncha;

– target simvoli aniqlanmaguncha;

– faylni oxiriga yetmaguncha.

Ostream sinfida quyidagi funksiyalar tavsiflangan: ostream& put(char C); ostream ga S simvolni joylashtiradi. ostream& write(const char* buffer,int size);

Buferda mavjudlarni ostreamga yozadi. Xatoga duch kelmaguncha yoki size simvollarni nusxasi olmaguncha simvollarni nusxasi olinadi. Bufer formatlanmasdan yoziladi. Nol simvollarga ishlov berish boshqa ishlov berishlardan farq qilmaydi. Quyidagi funksiya ishlov berilmagan (binar va matnli) ma’lumotlarni ostreamga uzatadi.

ostream& flush();

streambuf buferni olib tashlaydi.

Bu funksiyalardan tashqari istream sinfda >>, ostream sinfda esa << operatsiyalar qayta yuklangan. << va >> operatsiyalar ikkita operandga ega. Chap operandi – bu istream (ostream) sinfning obyekti, o‘ng operandi esa – bu dasturlash tilida ko‘rsatilgan tipdagi ma’lumot. Misol:

```
char ch, next, lookahead;
while ( cin.get( ch ))
{
  switch (ch) {
  case '/':
    //  izoxni peek() yordamida tekshirish
    // agar xa bo‘lsa satr qolganini o‘tkazish
    next = cin.peek();
    if ( next == '/' )
      cin.ignore( lineSize, '\n' );
    break;
  case '>':
    // leksemaga tekshirish >>=
    next = cin.peek();
    if ( next == '>' ) {
      lookahead = cin.get();
```

```
next = cin.peek();
if ( next != '=' )
   cin.putback( lookahead );
}
// ...
```

11.3. Formatlash

Ushbu ma’lumotlar uchun cout, cin, cerr, clog standart potoklarga kiritish << va chiqarish >> operatsiyalarni to‘g‘ridan to‘g‘ri qo‘llash qayta uzatish qiymatlarni tashki tavsiflash aytib o‘tilmagan formatlardan foydalanishga olib keladi.

Chiqaruvchi axborotni tavsiflash formatlari va ma’lumotlarni qiritishda qabul qilish qoidalari dasturlovchi orqali formatlash bayroqlari yordamida o‘zgartiriladi. Bu bayroqlar ios bazaviy sinfdagi hamma oqimlardan meros bo‘lgan. Formatlash bayroqlari aloxida qayd etilgan bitlar ko‘rinishida amalga oshirilgan va long x_flags sinfning protected komponentasida saqlanadi. Ularga murojaat etish uchun tegishli public funksiyalar mavjud.

Formatlash bayroqlardan tashqari ios sinfning kuydagi protected komponentalari ishlatiladi:

int x_width – chiqarish maydonning minimal yeni.

int x_precision – qiritishda xaqiqiy sonlarning tavsiflash aniqligi (kasr qisimning raqamlar soni);

int x_fill – chiqarishda to‘ldiruvchi simvol, probel – ko‘rsatilmagan holda.

Ushbu maydonlarni qiymatlarini olish (o‘rnatish) uchun quyidagi funksiyalar komponentalari ishlatiladi:

```
int width();
int width(int);
int precision();
int precision(int);
char fill();
char fill(char);
```

Yesli odnajdы vыbrat simvol-zapolnitel s pomoщyu *cout.fill*, on budet ostavatsya deystvitelnым, poka ne izmenitsya povtornым vыzovom *cout.fill.*

11.4. Manipulyatorlar

Manipulyatorlar - oqim ishini modifikatsiyalashini imkon etuvchi maxsus funksiyalar. Manipulyatorlarning xususiyati shundaki, ularni >> yoki << operatsiyalarning o‘ng operand sifatida foydalanish mumkin. Chap operand sifatida esa hardoimgidak oqim (oqimga ilova) ishlatiladi, va xudda shu oqimga manipulyator ta’sir etadi.

endl Vstavit simvol novoy stroki, zatem sbrosit bufer ostream

dec Pechatat v desyatichnoy sisteme (ustanovlen po umolchaniyu)

hex Pechatat v shestnadsaterichnoy sisteme

oct Pechatat v vosmerichnoy sisteme

ws Propuskat probelnыe simvolы

// shunday manipulyatorlar uchun #include <ionamip> talab etiladi

setfill(ch) ch simvol bilan bo‘sh joyni to‘ldirish

setprecision(n) n ga teng bo‘lgan suzuvchi nuqtali sonni chiqarish aniqliligini o‘rnatish

setw(w) w ga teng bo‘lgan qiritish yoki chiqarish maydonning enini o‘rnatish

setbase(b) b asosiga ega bo‘lgan butun sonlarning chiqarish

11.5. Oqimni xolati

Har bir oqim u bilan bog‘liq xolatga ega. Oqimni xolati enum o‘tkazish ko‘rinishida ios sinfida tavsiflanadi.

```
public:
enum io_state{
goodbit, //0X00 xatosi yo‘q
eofbit,  //0X01 faylni oxiri
```

failbit, //0X02 oxirgi operatsiya bajarilmagan

badbit, //0X04 mumkin bo‘lmagan operatsiyani ishlatishni harakat qilish

hardfail//0X08 taqdiriy xato

};

ios obyekti bilan oxirgi bajarilgan operatsiyasini natijalarini aniqlovchi bayroqlar state o‘zgaruvchisida mavjud. Shu o‘zgaruvchining qiymatlarini int rdstate() funksiyalari yordamida olish mumkin.

Bundan tashqari, oqimlar xolatini quyidagi funksiyalar orqali tekshirish mumkin:

int bad(); 1, agar badbit yoki hardfail

int eof(); 1, agar eofbit

int fail(); 1, agar failbit, badbit yoki hardfail

int good(); 1, agar goodbit

Agarda >> operatsiya ma’lumotlarni yangi tiplari uchun ishlatilsa, unda uni qayta yuklashda tegishli tekshirishlarni ko‘zda tutmoq lozim.

Funksiya-komponenta cout.fill va manipulyator setw ()

```
#include <iostream.h>
#include <iomanip.h>
void main(void) {
  cout << "Axborot jadvili " << endl;
  cout.fill (' . ');
  cout << "Kompaniya soxasi " << setw(20) << 10 << endl;
  cout << "Kompaniya daromadi va zarari " << setw(12) << 11 << endl;
  cout << "Kompaniya raxbariyati " << setw(14) << 13 << endl; }
```

11.6. Chiqarish operatorini qo‘shimcha yuklash

Chiqarish operatori ostream. sinfi obyektiga ilova qaytaruvchi binar operatordir. Umumiy holda qo‘shimcha yuklangan chiqarish operatori ta’rifi quyidagi ko‘rinishga ega:

ostream&

```
operator <<( ostream& os, const ClassType &object )
{
// obyektni tayyorlash uchun kod
// a’zolar xaqiqiy soni
os << // ...
// ostream obyekti qaytariladi
return os;
}
```

Bu ta’rif birinchi argumenti ostream obyektiga ilova ikkinchisi odatda konstant biror sinf obyektiga ilova. Qaytariluvchi qiymat ostream obyektiga ilova.

Birinchi argument ilova bo‘lgani uchun, chiqarish operatori sinf a’zosi sifatida emas oddiy funksiya sifatida ta’riflanishi zarur. Agar funksiya yopiq sinf a’zolariga murojaat qilishi zarur bo‘lsa do‘stona deb e’lon qilinishi zarur.

1. Chiqarish operatorini qo‘shimcha yuklash

```
class WordCount {
   friend ostream&
        operator<<( ostream&, const WordCount& );
public:
   WordCount( string word, int cnt=1 );
   // ...
private:
   string word;
   int occurs;
};
ostream& operator <<( ostream& os, const WordCount& wd )
{    // format: <hisoblagich> so‘z
    os << "< " << " > " > "
       << wd.word;
    return os;
    }
```

2. Yuklangan chiqarish operatoriga ega bo‘lgan sinfdan foydalanish.

```
#include <iostream>
#include "WordCount.h"
int main()
{   WordCount wd( "sadness", 12 );
  cout << "wd:\n" << wd << endl;
  return 0;}
```

Qo‘shimcha yuklangan chiqarish operatorini ofstream sinfi obyektlariga ham qo‘llash mumkin. Quyidagi misolda WordCount sinfi chiqarish operatori chaqiriladi:

```
#include <fstream>
#include "WordCount.h"
int main()
{    ofstream oFile( "word.out" );
   WordCount artist( "Renoir",12);
   oFile << artist;}
```

Nazorat uchun savollari

1. Qaysi sinflar asosida oqimlar bibliotekasi qurilgan?
2. Satrli oqimlarni va ularrning vazifalarini ko‘rsating.
3. Oqimlar usullarini ko‘rsating.
4. Formatlash uchun qanday komponenta funksiyalardan foydalanadi?
5. Manipulyatorlar vazifasini ko‘rsating.
6. Qaysi manipulyatorlar uchun #include <ionamip> ulash zarur?

12 bob. ISTISNOLAR

12.1. Istisno xolatlar

C++ tili OYD doirasida istisnolarga xizmat koʻrsatish standartini belgilab beradi. Istisno xolatlar (exception) dasturda xatoni – kutilmagan xodisani ifodalaydi. Dastur oʻzining ishlab chiqilishida koʻzda tutilmagan normal boʻlmagan vaziyatga duch kelganda, boshqaruvni ushbu muammoni xal qilishga qodir boʻlgan dasturning boshqa qismiga berish mumkin xamda yo dasturni bajarishni davom ettirish yoki ishni tugallash kerak. Istisnolarni joydan joyga tashlab berish (*yexcpletion throwing*) dasturning normal bajarilishiga toʻsqinlik qiladigan sabablarning tashxisi uchun foydali boʻlishi mumkin boʻlgan axborotni tashlab berish nuqtasida toʻplash imkonini beradi. Siz dastur tugallanishi oldidan zarur xatti-harakatlarni bajaradigan istisnolarga ishlov bergich (*exception handler*) ni aniqlashingiz mumkin. Dastur ichida yuzaga keladigan sinxron istisnolar deb nomlanuvchi istisnolarga xizmat koʻrsatiladi. Ctrl+C klavishalarini bosish kabi tashqi xolatlar istisno xisoblanmaydi.

Dasturda har bir istisno xolat sinf sifatida aniqlanadi. Masalan, quyida koʻrsatilan xolat fayllar bilan ishlash uchun uchta istisno xolatni aniqlaydi:

```
class file_open_error {};
class file_read_error {};
class file_write_error {};
```

Istisno xolatlar oʻzgaruvchilarni va sinf funksiya – elementlarini ishlatish mumkin. Har bir istisno xolat sinfga mos.

12.2. Istisnolarni qayta ishlash

Dastur istisno xolatni koʻrishdan va unga javob berishdan oldin istisno xolatni aniqlovchi C++dagi try operatorini ishlatish lozim. Istisnolarni generatsiya qila oladigan kod bloki try kalit-soʻz bilan boshlanadi va shakldor qavslar ichiga olinadi. Agar try blok ichida istisnoni topib olsa, dasturiy uzilish sodir boʻladi xamda quyidagi xatti-harakatlar ketma-ketligi bajariladi:

1.Dastur istisnoga ishlov bergichning toʻgʻri keladiganini qidiradi.

2.Agar ishlov bergich topilsa, stek tozalanadi va boshqaruv istisnolarga ishlov bergichga uzatiladi.

3.Agar ishlov bergich topilmagan bo‘lsa, ilovani tugatish uchun terminate funksiyasi chaqiriladi.

Yuzaga kelgan istisnoga ishlov beruvchi kod bloki catch kalit-so‘z bilan boshlanadi va shakldor qavs ichiga olinadi. Istisnoga ishlov bergichning kamida bitta kod bloki bevosita try blokining ortidan kelishi kerak. Dastur generatsiya qilishi mumkin bo‘lgan har bir istisno uchun o‘z ishlov bergichi ko‘zda tutilgan bo‘lishi kerak. Istisnolarga ishlov bergichlar navbatma-navbat ko‘rib chiqiladi xamda turi bo‘yicha catch operatoridagi argument (dalil) turiga to‘qg‘ri keladigan istisnoga ishlov bergich tanlab olinadi. Ishlov bergich tanasida goto operatorlari bo‘lmagan taqdirda, berilgan try bloki istisnolariga ishlov bergichning oxirgisidan keyin kelgan nuqtadan boshlab dasturning bajarilishi yana davom etadi.

Masalan, file_sopy funksiyani chaqirishda quyidagi try operatori istisno xolatni aniqlash imkonini beradi:

```
try
{    file_copy("SOURCE.TXT", "TARGET.TXT") ;
     };
```

Qanday istisno xolat ro‘y berganini aniqlash uchun try operatordan so‘ng dastur bitta yoki bir nechta catch operatorlarni joylashtirish lozim:

```
catch (file_open_error)
{
    cerr << "boshlangich yoki maqsadli faylni ochish xatoligi"<< endl;
    exit(1);
  }
```

Bu xolda xato tipiga qaramasdan kod xabardor qiladi va dasturni tugatadi. Agarda funksiyaning chaqiruvi xatosiz bajarilgan va istisno xatolar aniqlanmagan bo‘lsa C++ catch operatorini shunchaki etiborga olmaydi.

Qayta ishlovchilar tartibi muximdir.

```
try  {
```

```
            // ...
        }
    catch (ibuf) { // kiritish buferi to‘lishini qayta ishlash
                }
    catch (io) { // kiritish – chiqarish xatoligini qayta ishlash
            }
    catch (stdlib) { // bibliotekadagi istisno xolatni qayta ishlash
                }
    catch (...) {    // qolgan xamma istisnolarni qayta ishlash
            }
```

12.3. Istisnolarni generatsiya qilish

C++ o‘zi istisno xolatlarni yuzaga keltirmaydi. Ularni C++ ning throw operatoridan foydalangan dasturlar yuzaga keltiradi. Istisno yuzaga kelganda, throw operatoridagi <ifoda> nom berish ifodasi muvaqqat obyektni nomlaydi (initsiallashtiradi), Bunda muvaqqat obyektning turi ifoda argumenti (dalili) ning turiga mos keladi. Ushbu obyektning boshqa nusxlari, masalan, istisno obyektidan nusxa ko‘chirish konstruktori yordamida generatsiya qilinishi mumkin.

Masalan fayl ochilishida dastur xato kelib chiqish shartlarini tekshirish va throw file_open_error() istisno xolatni yuzaga keltirish mumkin.

Istisno xolatdan foydalanishda dastur xatoni kelib chiqish shartlarini tekshiradi va zarur bo‘lsa throw operatordan foydalangan xolda istisno xolatni yuzaga keltiradi. C++ throw operatorni uchratsa, u istisno xolatga tegishli ishlov beruvchini aktivizatsiya etadi (operatorlari istisno xolat sinfida aniqlangan funksiya. Istisno xolatga ishlov beruvchi funksiya bajarilgandan so‘ng C++ boshqarishni try operatordan keyin turgan birinchi operatorga qaytaradi. try operator esa istisno xolatni aniqlashini xal etuvchi operator. Keyin esa catch operatordan foydalanib dastur kanday istisno xolat ro‘y berganini aniqlaydi va tegishli xolda javob beradi.

12.4. Kutilmagan istisnolarni qayta ishlash

Agar dasturda kuzda tutilmagan istisno xodisa yuz bersa standart istisnolarni kayta ishlovchi ishlatiladi. Kup xollarda bu standart kayta ishlovchi dastur bajarilishini tuxtatib kuyadi.

Avval unexpected funksiyasi chaqirilib, undan so‘ng ko‘zda tutilgan bo‘yicha terminate funksiyasi ishga tushadi. Bu funksiya dasturni to‘xtatish uchun abort funksiyasini chaqiradi.

Dasturda maxsus kayta ishlovchidan foydalanish uchun set_unexpected va set_terminate funksiyasidan foydjalanish lozim. Bu funksiyalar prototiplari except.h sarlavxali faylda aniklangan. Bu funksiyalar void tipiga ega bo‘lib parametrlarga ega bo‘lmasligi kerak.

12.5. Istisno xolatning ma’lumotlar elementlaridan foydalanish

Yuqorida ko‘rib o‘tilgan misollarda dastur, catch operatordan foydalanib, qanday istisno xolat ro‘y berganini va ularga tegishli xolda javob berishini imkonini beradi. Masalan, file_open_error istisno xolatda dastur xatoni chaqiruvchi fayl nomini bilish lozim. Istisno xolatga tegishli shunday ma’lumotni saqlash uchun dastur istisno xolat sinfiga ma’lumotlar elementlarini qo‘shish. Agar keyinchalik dastur istisno xolatni yuzaga keltirsa, u ushbu ma’lumotni, quyida ko‘rsatilgandek, istisno xolatiga ishlov beruvchi funksiyaga o‘zgaruvchi sifatida uzatadi:

throw file_open_error(source);
throw file_read_error(344);

Istisno xolatga ishlov berishda bu parametrlar sinfga tegishli o‘zgaruvchilarga o‘zlashtirilishi mumkin (konstruktorga o‘xshaydi). Masalan, sinfning tegishli o‘zgaruvchisiga xatoga yo‘l qo‘ygan faylni ismini o‘zlashtirish uchun quyidagi operatorlar file_open_error istisno xolatni o‘zgartiradi:

class file_open_error

```
{ public:
file_open_error(char *filename) { strcpy(file_open_error::filename, filename); }
char filename[255] ;  };
```

12.6. Istisno xolatlar va sinflar

Sinf yaratishda shu sinfga tegishli istisno xolatlarni aniqlash mumkin. Aniq sinfga tegishli istisno xolatni yaratish uchun ushbu istisno xolatni sinfning umumiy (public) elementlari sifatida kiritish zarur.

Masalan diapazon chegarasidan chiquvchi indeks qiymatini bilish zarur bo‘lsin:

```
class Vector {
  // ...
public:
  class Range {
  public:
    int index;
    Range(int i) : index(i) { }
  };
  // ...
  int& operator[](int i)
  // ...
        };
      int Vector::operator[](int i)
        {
if (o<=i && i <sz) return p[i];
throw Range(i);
        }
```

Mumkin bo‘lmagan indeks qiymatini bilish uchun istisno xolatni tasvirlovchi obyektga nom berish kerak:

```
void  f(Vector& v)
```

```
{
    // ...
  try {
    do_something(v);
      }
     catch (Vector::Range r ) {
     cerr << "mumkin bo‘lmagan indeks" << r.index << '\n';
     // ...
  }
  // ...
}
```

Qavsdagi konstruksiya tavsif bo‘lib funksiya formal parametriga mosdir. Unda parametr tipi va yuzag kelgn istisno nomi berilishi mumkin.

12.7. Istisnolar va konstruktorlar

Istisnolar konstruktordagi xatolar xaqida ma’lumot berishga imkon beradi. Konstruktor chaqiruvchi funksiya tekshirib ko‘rishi mumkin bo‘lgan qiymat qaytarmagani uchun istisnolarsiz quyidagicha xatolik xaqida ma’lumot berish mumkin:

1. Obyektni xatolik bilan qaytarish toki foydalanuvchi o‘zi tekshirib ko‘rsin.
2. Lokal bo‘lmagan o‘zgaruvchiga obyekt yaratilmagani xaqida ma’lumot beruvchi qiymat o‘rnatish.

Istisnolar obyekt yaratilmagani xaqidagi ma’lumotni tashqariga uzatishga imkon beradi:

```
Vector::Vector(int size)
{
  if (sz<0 || max<sz) throw Size();
  // ...
```

```
}
```

Vektor yaratilayotgan funksiyada noto‘g‘ri o‘lcham (Size()) xatoligini qayta ishlash mumkin:

```
Vector* f(int i)
{
  Vector* p;
  try {
    p = new Vector v(i);
  }
  catch (Vector::Size) {
    // vektor noto‘qri o‘lchami
  }
  // ...
  return p;
}
```

Nazorat uchun savollar

1. Istisnolarni nima uchun generatsiya va qayta ishlash kerak?
2. Istisno generatsiyasi sintaksisini keltiring.
3. Istisnoni qayta ishlash sintaksisini keltiring.
4. Istisnolar bilan qanday operatorlar bog‘liq?
5. Istisnoni generatsiya qiluvchi funksiya sintaksisini keltiring.

13 bob. STRANDART SHABLON SINFLAR BIBLIOTEKASI

13.1. STL tarkibi. Asosiy konteynerlar

Biblioteka yadrosi uchta elementdan iborat: konteynerlar, algoritmlar va iteratorlar. Konteynerlar (containers) – bu boshqa elementlarni saqlovchi obyektlar. Masalan, vektor, chiziqli ro‘yxat, to‘plam. Assotsiativ konteynerlar (associative containers) kalitlar yordamida ularda saqlanadigan qiymatlarni tezkor olish imkonini yaratadi. Har bir sinf – konteynerida ular bilan ishlash uchun mo‘ljallangan funksiyalar to‘plami aniqlangan. Masalan, ruyxat elementlarni kiritish, chiqarish, va qo‘shish funksiyalarni o‘z ichiga oladi. Algoritmlar (algorithms) konteyner ichidagilar ustidan operatsiyalar bajaradi. Konteyner ichidagilarni initsializatsiyalash, qidirish, saralash va almashtirish uchun algoritmlar mavjud. Ko‘p algoritmlar konteyner ichidagi elementlarni chiziqi ro‘yxatini ifodalaydovchi ketma-ketlik (sequence) bilan ishlash uchun mo‘ljallangan. Iteratorlar (iterators) – bu konteynerga nisbatan ko‘rsatkich sifatida bo‘lgan obyektlar. Ular massiv elementlariga ruxsat oluvchi ko‘rsatkichlar kabi, konteyner ichidagiga ruxsat olish imkoni beradi.

vector <vector.h> dinamk massiv

list <list.h> chiziqli ro‘yxat

deque <deque.h> ikki tarafli dvustoronnyaya tartib

set <set.h> to‘plam

multiset <set.h> har bir elementi noyob bo‘lishi shart emas to‘plam

map <map.h> kalit/ qiymat juftlikni saqlash uchun assotsiativ ro‘yxat. Bunda har bir kalit bitta qiymat bilan bog‘langan.

multimap <map.h> har bir kalit bilan ikkita yoki ko‘proq qiymatlar bog‘langan

stack <stack.h> stek

queue <queue.h> tartib

priority_queue <queue.h> birinchi o‘rindagi tartib

13.2. Konstruktorlar. Iteratorlar

Ixtiyoriy sinf-konteyner ko‘rsatilmagan holda konstruktor va destruktorni nusxalovchi konstruktorga ega. Masalan, vektor sinf-konteynerning konstruktori va destruktori:

Ixtiyoriy obyekt uchun ko‘rsatilmagan holda konteynerda saqlanuvchi konstruktor mavjud bo‘lishi shart. Undan tashqari, obyekt uchun < va == operatorlar aniqlanish lozim.

Iteratorlar bilan ko‘rsatkichlar kabi ishlash mumkin. Ularga *, inkrement, dekrement operatorlarni qo‘llash mumkin. Iterator tipi sifatida har xil konteynerlarda aniqlangan iterator tip elon qilinadi.

Iteratorlarning beshta tipi mavjud:

1. Kiritish iteratorlar (input_iterator) tenglik, nomini o‘zgartirish va inkrementa operatsiyalarni qo‘llaydi.

==, !=, *i, ++i, i++, *i++

Kiritish iteratsiyasining maxsus xolati istream_iterator iborat.

2. Chiqarish iteratorlar (output_iterator) o‘zlashtirish operatorning chap tarafidan imkon bo‘lgan isimning o‘zgartirish va inkrementa operatsiyalar qo‘llanadi.

++i, i++, *i=t, *i++=t

Chiqarish iteratsiyasining maxsus xolati ostream_iterator.

3. Bitta yo‘nalishdagi iteratorlar (forward_iterator) kiritish/chiqarish operatsiyalarning barchasini qo‘llaydi, bundan tashqari chegarasiz o‘zlashtirishning imkonini beradi.

==, !=, =, *i, ++i, i++, *i++

4. Ikki yo‘nalishdagi iteratorlar (biderectional_iterator) forward-iteratorlarning barcha xususiyatlariga ega, bundan tashqari, konteynerni ikkita yo‘nalishi bo‘yicha o‘tish imkonini beradigan qo‘shimcha dekrementa (--i, i--, *i--) operatsiyasiga ega.

5. Ixtiyoriy ruxsatga ega bo‘lgan iteratorlar (random_access_iterator) biderectional-iteratorlarning barcha xususiyatlariga ega, bundan tashqari solishtirish va manzil arifmetikasi operatsiyalarni qo‘llaydi.

i+=n, i+n, i-=n, i-n, i1-i2, i[n], i1<i2, i1<=i2, i1>i2, i1>=i2

SHuningdek, STLda teskari iteratorlar (reverse iterators) qo‘llaniladi. Ketma-ketlikni teskari yo‘nalishda o‘tuvchi ikki yo‘nalishli yoki ixtiyoriy ruxsatga ega bo‘lgan iteratorlar teskari iteratoralar bo‘lishi mumkin.

13.3. Xotirani taqsimlovchilar, predikatlar va solishtirish funksiyalari

Konteynerlarga, algoritmlarga va STLdagi iteratorlarga qo‘shimcha bir nechta standart komponentalar ham qo‘llaniladi. Ulardan asoslari esa xotira taqsimlovchilar, predikatlar,va solishtirish funksiyalaridir. Har bir konteynerda uning uchun aniqlangan va konteyner uchun xotirani belgilash jarayonini boshqaradigan xotira taqsimlovchisi (allocator) mavjud. Ko‘rsatilmagan holda esa xotira taqsimlovchisi allocator sinf obyektidir. Xususiy taqsimlovchini tavsiflash mumkin. Ba’zi bir algoritmlar va konteynerlarda muxim tipdagi predikat ataluvchi funksiyalar ishlatiladi. Predikatlar unar va binar bo‘lishi mumkin. U yoki bu qiymatni olish aniq shartlari dasturchi orqali aniqlanadi. Unar predikatlarning tipi – UnPred, binar predikatlarning esa - BinPred. Argumentlar tipi konteynerda saqlanuvchi obyektlar tipiga mos.

Ikta elementlarni solishtirish uchun binar predikatlarning maxsus tipi aniqlangan. U solishtirish funksiya (comparison function) deyiladi. Agarda birinchi element ikinchidan kichik bo‘lsa, unda funksiya rost qiymatni qaytaradi. Comp tip funksiya tipidir.

STL da obyekt-funksiyalar o‘ziga xos ahamiyatga ega.

Obyekt-funksiyalar – bu sinfda «kichik qavslar» () operatsiyasi aniqlangan sinf nusxalari. Ba’zi bir xollarda funksiyalarni obyekt-funksiyalarga almashtirish qulay deb hisoblanadi. Obyekt-funksiya funksiya sifatida ishlatilsa, unda uni chaqirish uchun operator () operator ishlatiladi.

13.4. Vector-vektor konteynerlari

STL da vector vektor dinamik massiv sifatida aniqlanadi. Massiv elementlariga indeks orqali ruxsat beriladi. Vector sinfida quyidagi konstruktorlar aniqlangan:

Birinchi shakl bo‘sh vektor konstruktorini tavsiflaydi.

Konstruktor vektorning ikkinchi shaklida elementlar soni – bu son, har bir elementi esa qiymat qiymatiga teng. Qiymat parametri ko‘rsatilmagan holdagi qiymat bo‘lishi mumkin.

Konstruktor vektorning uchinchi shakli – bu nusxalash konstruktori.

To‘rtinchi shakli – bosh va oxirgi iteratorlar orqali elementlar diapazonini o‘z ichiga olgan konstruktor vektor.

Vektorda saqlanadigan ixtiyoriy obyekt uchun ko‘rsatilmagan holda konstruktor aniqlash zarur. Bundan tashqari, obyekt uchun < va == operatorlar aniqlanishi lozim.

Vektor sinfi uchun quyidagi solishtirish operatorlari mavjud:

==, <, <=, !=, >, >=.

Bundan tashqari, vector sinf uchun [ ] indeks operatori aniqlangan.

13.5. Ikki yo‘nalishli tartib. Ro‘yxat. Assotsiativ konteynerlar

deque – vektor kabi, ixtiyoriy ruxsat iteratorlarni qo‘llovchi ketma-ketlik ko‘rinishi. Bundan tashqari, u o‘zgarmas vaqtda boshida yoki oxirida kiritish va ochirish operatsiyalarni qo‘llaydi. O‘rtada kiritish va o‘chirish chiziqli vaqtni egallaydi. Xotirani boshqarishiga ishlov berish esa vektorlar kabi avtomatik ravishda bajariladi.

Ruyxat – ikki yo‘nalishli iteratorlarni qo‘llaydigan hamda kiritish va o‘chirish operatsiyalarni o‘zgarmas vaqtda ketma-ketlikni ixtiyoriy joyida bajaradigan, shuningdek, xotirani boshqarishiga avtomatik ravishda ishlov beruvchi ketma-ketlik ko‘rinishi. Vektorlar va ikkitarafli tartiblardan farqi shundaki elementlar ro‘yxatiga tez va ixtiyoriy ro‘xsat qo‘llanmaydi, lekin ko‘pgina algoritmlarga esa ketma-ketlik ruxsat zarur.

Assotsiativ massiv juft qiymatlardan iborat. (key) kalit deb atalgan bitta qiymatni bilib (mapped value) aks etuvchi qiymat deb atalgan ikkinchi qiymatga ruxsat olishimiz mumkin.

Assotsiativ massivni massiv indekslari butun tiplardan iborat bo‘lmagan massiv sifatida tavsiflash mumkin:

V& operator[](const K&) K ga mos keluvchi V ga ilovani qaytaradi.

Assotsiativ konteynerlar – bu assotsiativ massivning umumiy tushunchasi.

map assotsiativ konteyner − bu kalit yordamida qiymatga tez ega bo‘lish imkonini yaratadigan juftlik (kalit, qiymat) ketma-ketligi. map konteyneri ikki yo‘nalishli iteratorni tavsif etadi.

map assotsiativ konteyneri kalit tiplari uchun “<” operatsiyasi mavjudligini talab qiladi. U kalit bo‘yicha saralangan o‘z elementlarini saqlaydi. Saralash almashuvi esa tartib bo‘yicha bajariladi.

map sinfida quyidagi knstruktorlar aniqlangan:

birinchi shakli bo‘sh assotsiativ konteynerning konstruktorini tavsiflaydi. Ikkinchi shakli – konstruktor nusxasi, uchinchisi – elementlar diapazonini kamrab olgan assotsiativ konteynerning konstruktori.

O‘zgartirish operatsiyasi aniqlangan:

map& operator=(const map&);

quyidagi operatsiyalar aniqlangan: ==, <, <=, !=, >, >=.

mapda kalit/qiymat juftliklar pair tiplagi obyektlar ko‘rinishida saqlanadi.

Kalit/qiymat juftliklarni faqatgina pair sinf konstruktorlari yordamida, balki pair tipdagi obyektlarni yaratuvchi va ma’lumotlar tiplaridan parametrlar sifatida foydalanuvchi make_pair funksiya yordamida yaratish mumkin.

Assotsiativ konteyner uchun o‘ziga xos operatsiya – bu ([]) indeksatsiyalash operatsiyasi yordamida assotsiativ qidiruv.

mapped_type& operator[](const key_type& K);

set to‘plamini assotsiativ massiv sifatida ko‘rish mumkin. Unda qiymatlar ahamiyatga ega emas, shuning uchun faqat kalitlarni ko‘zatish mumkin. To‘plamlar assotsiativ massiv kabi T tip uchun (<) “kichik” operatsiyani mavjudligini talab etadi. U o‘z elementlarini saralangan holda saqlaydi. saralash almashuvi esa tartibda bajariladi.

Nazorat uchun savollar

1. Biblioteka yadrosi qanday elementlardan iborat?
2. Har qanday konteyner qanday konstruktorlarga ega?
3. Iteratorlar tiplarini ko‘rsating.
4. Assotsiativ massivlar qanday xususiyatlarga ega?
5. Elementlarga murojaat usullarini ko‘rsating.
6. Elementlarni o‘chirish usullarini ko‘rsating.
7. Konteyner hajmini o‘zgartirish uchun qanday usuldan foydalaniladi?

14 bob. XODISAVIY BOSHQARILUVCHI DASTURLASH

14.1. Komponentlar. Komponentli sinflarni e’lon qilish

C++Builder, nafaqat ANSI C++ standarti kiritayotgan yangiliklarni qo‘llab-quvvatlaydi, balki tilni yangi imkoniyatlar bilan boyitadi. Shuni tushunib olish muximki, tilni kengaytirish xech qachon quruq maqsad bo‘lib qolmagan, va hamon standart C++ doirasida yozilgan mantlarni kompilyatsiya qilish mumkin. Biroq ilovalarni tez ishlab chiqish texnologiyasi (RAD) uchun C++Builder taqdim etgan imtiyozlardan to‘liq foydalanish uchun, kiritilgan til kengaytirishlarni qabul qilishga to‘g‘ri keladi. Kengaytirishlarning ayrimlari (maslan, _classid) ni C++Builder asosan ichki foydalanish uchun rezervlaydi. C++ning eng ahamiyatli kengaytirishlari asosan tarkibli sinflarga mansub bo‘lib, C++Builder muxitida ishlab chiqilayotgan ilovalarda muttasil uchrab turadi. Komponentlar ko‘p o‘rinda, C++standart sinflariga qaraganda, yuqoriroq darajadagi Inkapsulyatsiyalashga erishadilar. Buni tugmachaga ega bo‘lgan dialogni ishlab chiqish kabi oddiy misolda ko‘rib chiqamiz. Windows uchun namunaviy C++dasturida tugmachani «sichqoncha» bilan bosish natijasida WM_LBUTTONDOWN xabarining generatsiyasi sodir bo‘ladi. Bu xabarni dastur yo switch operatorida, yoki chaqiriqlar jadvali (RESPONCE_TABLE) ning tegishli satrida «tutib olish»i, keyin esa ushbu xabarga javob protsedurasiga uzatishi kerak.

C++Builder o‘zlashtirilishi qiyin bo‘lgan bu kabi dasturlash o‘yinlariga chek qo‘ydi. Komponenta tugmachasi avvaldanoq unga OnClick voqeasi bilan bosishga javob beradigan qilib dasturlangan. Bu o‘rinda talab qilinayotgan narsa - tayyor metodni tanlab olish (yoki o‘zinikini yozish) hamda Obyektlar Inspektori yordamida berilgan voqea-xodisaga ishlov bergichga kiritish. C++Builder tarkibiga kiradigan Vizual Komponentalar Kutubxonasi - VCL sinflarining ilgarilovchi e’lonlari _declspec modifikatoridan foydalanadi:

_declspec(<spetsifikator>)

Bu kalit-so‘z, nafaqat bevosita modifikatsiyalanayotgan e’lon oldidan, balki e’lonlar ro‘yxatining to‘g‘ri kelgan yerida paydo bo‘lishi mumkin, bunda spetsifikator quyidagi qiymatlardan birini qabul qiladi:

delphiclass - u TObject sinfiga tegishli VCL ning bevosita yoki bilvosita xosilalarining ilgarilovchi e’loni uchun qo‘llanadi. U VCL ning RTTI ,konstruktorlar, destruktor va istisnolar bilan muomalasida muvofiqlik qoidalarini belgilaydi.

delphireturn - u Currency, AnsiString, Variant, TDateTime va Set sinflariga tegishli VCL ning bevosita yoki bilvosita xosilalarining ilgarilovchi e’loni uchun qo‘llanadi. U VCL ning parametrlar va a’zoqfunksiyalarning qaytarilayotgan qiymatlari bilan muomalasida muvofiqlik qoidalarini belgilaydi. Pascal implementation tarkibli sinf Obyektli Pascal tilida ishga tushirilganini ko‘rsatadi.

VCL sinf quyidagi cheklanishlarga ega:

- Virtual bazaviy sinflarga vorislik qilish man etilgan.
- Tarkibli sinflarning o‘zlari vorislik uchun bazaviy sinf sifatida xizmat qila olmaydi.
- Tarkibli obyektlar uyumning dinamik xotirasida new operatori yordamida yaratiladi.

14.2. Xususiyatlarni e’lon qilish. Voqealar ishlatgichlarining e’lonlari

C++Builder tarkibli sinflar xususityalarini identifikatsiya qilish uchun _property modifikatoridan foydalanadi. Xususiyatni tavsiflash sintaksisi quyidagi ko‘rinishga ega:

property<xususiyat turi > <xususiyat nomi >={<atributlar ro‘yxati >};

bu yerda atributlar ro‘yxati quyidagi xususiyatlar atributlarining sanog‘iga ega:

write=<ma’lumotlar a’zosi yoki yozuv metodi > ma’lumotlar a’zosiga qiymat berish usulini aniqlaydi;

read=<ma’lumotlar a’zosi yoki o‘qish metodi > ma’lumotlar a’zosining qiymatini olish usulini aniqlaydi;

default=<bul konstatntasi > .dim kengayishli shaklga ega bo‘lgan yashirin xususiyatlar qiymatini saqlashni ruxsat beradi yoki man etadi;

stored=<bul konstantasi yoki funksiya > .dfm. kengayishli shaklga ega bo‘lgan faylda xususiyat qiymatini saqlash usulini aniqlaydi.

C++Builder ilovani loyixalash bosqichida Obyektlar Inspektori tomonidan aks ettiriladigan komponentalar xususiyatlarini spetsifikatsiyalash uchun _published modifikatoridan foydalanadi. Agar komponentaning ishlab chiquvchisi biron-bir xususiyat qiymatini modifikatsiyalashga ruxsat berishni xoxlab qolsa, bu xususiyat _ published sifatida e'lon qilinmaydi. Ushbu kalit-so'z bilan aniqlanayotgan ko'rimlilik qoidalari public sifatida e'lon qilingan ma'lumotlar a'zolari, metodlar va xususiyatlarning ko'rimlilik qoidalaridan farq qilmaydi. Yagona farq shundaki, dasturning ishlash paytida Obyektlar Inspektoriga RTTI axboroti uzatiladi.

C++Builder voqealar ishlatgichlari funksiyalarining e'loni uchun _closure modifikatoridan foydalanadilar:

<tur>(_closure*<name>)(<pametrlar ro'yxati >)

Bu kalit-so'z funksiya ko'rsatkichini name nomi bilan aniqlaydi. Oddiy funksiyaning 4 baytli adresli ko'rsatkichidan farqli o'laroq (bu ko'rsatkich CS:IP kod registrlariga uzatiladi), 8 baytli _closure yana yashirin parametrni ham uzatadi (joriy sinf ekzemplyariga txis o'zgaruvchan ko'rsatkichi).

8 baytli ko'rsatkichlarning kiritilishi, nafaqat aniqlangan sinfning biron-bir funksiyasini chaqirib olish imkonini beradi, balki ushbu sinfning aniqlangan ekzemplyaridagi funksiyaga murojaat qilish imkonini ham beradi. Bu qobiliyat Obyektli Paskaldan o'zlashtirilgan edi, _closure yesa Vizual Komponentalar Kutubxonasidagi voqealar mexanizmini ishga tushirishda xavodek zarur bo'lib qoldi.

14.3. Funksiyalarning tez chaqirilishi. Nomlar fazosi

Parametrlari protsessorli registrlar orqali uzatiladigan funksiyalarni e'lon qilishda _fastcall modifikatori qo'llanadi:

<qaytarilayotgan tur >_fastcall<name>(<parametrlar ro'yxati >)

Bu kalit-so'z name nomli dastlabki uchta turlashtirilgan parametr (ro'yxat bo'yicha chapdan o'ngga) stek orqali emas, balki AX, BX va DX protsessorli registrlar orqali uzatilishini aniqlaydi. Agar parametr qiymati registrga sig'masa, ya'ni parametr orqali suzuvchi nuqtali sonlarni, tuzilmalar va funksiyalarni uzatishda, u qo'llanmaydi,

Xolisanillo aytganda, funksiyalarning tez chaqirilishi C++Builder kompilyatorininggina vazifasiga kirmaydi. Voqealarga ishlov berish funksiyalarini e’lon qilishda_fastcall ning qo‘llanishiga aloxida e’tibor berish kerak. Bu voqealarni C++Builder avtomatik tarzda generatsiya qiladi.

Oddiy ilovalarning ko‘pi dastlabki dastur matniga ega bo‘lgan bir nechta fayldan iborat. Bu fayllar dasturchilar guruxi tomonidan yaratilishi va xizmat ko‘rsatilishi mumkin. Pirovard natijada barcha fayllar birga to‘planadi va tayyor ilovani yig‘ishdan iborat bo‘lgan so‘nggi protseduradan o‘tadi.

An’anaviy tarzda qabul qilinishicha, biron bir lokal soxa (funksiya, sinf tanasi yokitranslyatsiya moduli) ga kiritilmagan barcha nomlar umumiy global ismlarni bo‘lib olishadi. Shuning uchun, agar ayrim modullarni yig‘ish jarayonida nomlar takroran aniqlangani ayon bo‘lib qolsa, bu holda har bir nomni qandaydir yo‘l bilan farqlash zarurligini talab qiladi. C++da bu muammoning yechilishi nomlar fazosi(namespace) mexanizmi zimmasiga yuklatilgan.

Bu mexanizm ilovani bir necha tarmoq tizimlar (tizimchalar) ga bo‘lib tashlash imkonini beradi, bunda har bir tarmoq tizim nomlarni tanlashda erkin ish tutadi, hamda uning muallifi xuddi shunday ismlardan biron boshqa kimsa foydalanishi mumkinligiga qayg‘urmasa ham bo‘ladi. Har bir tarmoq tizim global nomlar umumiy fazosida o‘zining paydo bo‘lganini namespace kalit-so‘zdan keyin kelgan unikal identifikator yordamida identifikatsiya qiladi:

namespace<identifikator> {[<e’lon qilish >]}

Identifikatsiya qilingan nomlar fazosi elementlariga kirishning uchta usuli mavjud:

Konkret elementga ochiq-oydin kirish kvalifikatsiyasi:

ALPHA :: vart;//ALPHA BETA::F1dagi o‘zgaruvchiga kirish; //BETA dagi o‘zgaruvchiga kirish

Barcha elementlarga kirish:

using namespace::ALPHA;//ALPHA dagi barcha nomlarga kirish

Nomlarning lokal fazosida yangi identifikatorning e’lon qilinishi:

using :: new_name;//identifikatorning qo‘shilishi

14.4. Ochiq-oydin e’lonlar. O‘zgaruvchan e’lonlar

Odatda bitta parametrli konstruktor e’lon qilingan sinf obyektlariga turlari avtomatik tarzda (yashirish holda) o‘z sinfi turiga qayta o‘zgaradigan qiymatlarni berish mumkin. Konstruktorni e’lon qilishda explicit konstruktoridan foydalanish mumkin:

explicit<konstruktorni e’lon qilish >

Bu holda berilgan sinf konstruktorlarini explicit kalit-so‘z bilan e’lon qilishda sinfning barcha obyektlariga faqat shunday qiymatlarni berish mumkinki, bu qiymatlar turlari o‘z sinfi turiga ochiq-oydin qayta o‘zagaradigan bo‘lishi kerak.

```
class X
public:
explicit X(int);
{explicit X(const char*, int =0);
};
void f(X arg)
{ X a = X (1) ;
  X b = X("satr",0);
  a = X(2);  }
```

Konstruktorlarning ochiq-oydin e’lonlari shuni talab qiladiki, nom berish operatorlaridagi qiymatlar qaysi sinfiy tur obyektlariga berilgan bo‘lsa, ular xuddi shu sinfiy turga qayta o‘zgartirilishini talab qiladi. Fon masalasi, uzish ishlatgichi yoki kiritish-chiqarish porti tomonidan o‘zgartirilishi mumkin bo‘lgan o‘zgaruvchini e’lon qilishda volatile modifikatori qo‘llanadi:

volatile<tur><obyekt nomi>;

C++da volatile kalit-so‘zning qo‘llanishi sinflar va a’zo-funksiyalarga ham tegishlidir. Bu kalit-so‘z ko‘rsatilgan obyekt qiymatiga nisbatan taxminlar qilishni kompilyatorga ta’qiqlaydi, chunki bunday qilinsa, ushbu obyektni o‘z ichiga olgan ifodalarni hisoblashda, uning qiymati har bir daqiqada o‘zgarib ketishi mumkin. Bundan tashqari o‘zgarib turadigan o‘zgaruvchi register modifikatori bilan e’lon

qilinishi mumkin emas. Quyidagi misol ticks o‘zgaruvchisini vaqtli uzilishlar qayta ishlagichi modifikatsiya qiladigan taymerni ishga tushirishga misol bo‘la oladi.

```
volatile int ticks;
void timer( ) // Taymer funksiyasini e’lon qilish
tickC++;
void wait (int interval)
{
ticks =0;  while (ticks < interval); // Kutish sikli
}
```

Aytaylik, uzilishni qayta ishlatgichi - timer real vaqt soatidagi apparat uzilishi bilan tegishli tarzda assotsiatsiya qilindi. ticks o‘zgaruvchisining qiymati ushbu qiymat parametri tomonidan berilgan vaqt intervaliga teng kelmaguncha, wait protsedurasi kutish siklini ishlataveradi. C++kompilyatori sikl ichidagi har bir qiyoslash oldidan volatile ticks o‘zgaruvchisining qiymatini, sikl ichidagi o‘zgaruvchining qiymati o‘zgarmaganiga qaramay, ortiqcha yuklashi lozim. Ayrim optimallashtiruvchi kompilyatorlar bunday «xavfli»xatoga yo‘l qo‘yishlari mumkin.

Xatto konstantali ifodaga kirganiga qaramay o‘zgartirilishi mumkin bo‘lgan o‘zgaruvchan o‘zgaruvchining boshqa bir turi mutable modifikatori yordamida e’lon qilinadi:

mutable<o‘zgaruvchining nomi>;

mutable kalit-so‘zning vazifasi shundan iboratki, u biron-bir sinf ma’lumotlari a’zolarini spetsifikatsiya qiladi, bunda ushbu ma’lumotlar a’zolari mana shu sinfning konstantali funksiyalari tomonidan modifikatsiya qilinishi mumkin bo‘lishi kerak. Ma’lumotlar a’zosi count ni F1 konstantali funksiya modifikatsiya qiladigan misolni ko‘rib chiqaylik:

```
class A{
public: mutable int count; int F1 (int p=0)const// F1 funksiyasini e’lon qilish
count=p++return count;//PI count ni qaytarib beradi
}
void vain(){
```

A, a;

Cout<<a.F1(3)<<endi; //main 4 qiymatini beradi

Nazorat uchun savollar

1. Komponentalarning standart sinflardan farqi nimadan iborat?
2. Xususiyatlar qanday e’lon qilinadi?
3. Voqealar ishlatgichlar qanday e’lon qilinadi?
4. Nomlar fazosi mexanizmidan foydalanish.
5. Ochiq oydin va o‘zgaruvchan e’lonlar nima uchun ishlatiladi?

15 bob. VORISLIKDAN FOYDALANISH XUSUSIYATLARI. INTERFEYSLAR

15.1. Object: global supersinf. equals va toString usullari

Object sinfi hamma sinflar ajdodi hisoblanadi. Java tilida xar bir sinf Object sinfini kengaytiradi. Lekin class Employee extends Oblects yozish shart emas. Agar supersinf oshkor koʻrsatilmagan boʻlsa Object supersinf hisoblanadi. Java tilida xar bir sinf Object sinfini kengaytirgani uchun Object sinfi imkoniyatlarini bilish muximdir.

Object tipidagi oʻzgaruvchini ixtiyoriy tipdagi obyektga ilova sifatida ishlatish mumkin:

Object obj = new Employee("Garri Xaker", 35000);

Bu tipdagi oʻzgaruvchidan foydalanish uchun avval boshlangʻich tipni aniqlab, tiplarni keltirishni amalga oshirish lozim:

Employee ye = (Employee) obj;

Object sinfining equals usuli ikki obyekt bir xilligini tekshiradi. Lekin equals usuli Object sinfiga tegishli boʻlgani uchun, ikkalasi bir xotira qismiga ilova qilganligini tekshiradi. Ikki obyekt ekvivalentligini tekshirish uchun equals usulini qoʻshimcha yuklash lozim. Mukammal equals usuli yaratish qoidalari.

1. Oshkor otherObject parametrini chaqirish — keyinchalik uning tipini other deb atalgan boshqa oʻzgaruvchi tipiga keltirish lozim.

2. Tekshirish, this va otherObject ilovalar bir xilmi:

i f (this == otherObject) return true;

Odatda obyektlar maydonini solishtirgandan koʻra ilovalarni solishtirish osondir.

3. Tekshirish otherObject ilova nulga (null) tengmi. Agar xa boʻlsa false qiymat qaytarish. Bu tekshirishni albatta amalga oshirish lozim.

if (otherObject == null) return false;

4. Tekshirish this va other obyektlari bitta sinfga tegishlimi.

Bu tekshirish "simmetriklik qoidasiga " koʻra majburiydir.

if (getClass() != otherObject.getClass()) return false;

5. Talab qilingan sinf oʻzgaruvchisiga otherObject obyektini oʻzgartirish:

ClassName other = (ClassName)otherObject;

6. Xamma maydonlarni solishtirish. Asosiy tipdagi maydonlar uchun == operatori, obyektli maydonlar uchun —equals usuli qo‘llanadi. Agar ikki obyekt hamma mydonlari bir xil bo‘lsa true qaytarish, aks holda false.

return fieldl == other.fieldl

&& field.2. equals (other . field2)

Masalan.

class Employee{

public boolean equals(Object otherObject) {

// Obyektlarni tez solishtirish,

if (this == otherObject) return true;

// Agar oshkor parametr — null, false qiymat qaytaradi,

if (otherObject == null) return false;

// Agar sinflar ustma ust tushmasa, ular ekvivalent emas.

if (getClass () != otherObject.getClassO) return false;

// Obyekt otherObject tipi Employee va u nulga teng emas.

Employee other = (Employee) otherObject;

// Obyektlar maydonlarini solishtirish,

return name.equals(other.name)

&& salary = other.salary

&& hireDay.equals(other.hireDay);

}

}

Obyekt tipini getClass usuli orqali aniqlanadi. Obyektlar o‘zaro teng bo‘lishi uchun bir sinf obyektlari bo‘lishlari kerak.

Voris ichida avvval spersinf equals usulini chaqirish lozim. Agar bu tekshirish false qiymat qaytarsa, demak obyektlar teng emas. Agar tekshirish muvaffaqiyatli bajarilsa ostki sinf maydonlarini tekshirishga o‘tish mumkin.

Masalan quyidagicha.

class Manager extends Employee

```
{
public boolean.equals(Object otherObject)
{
if (!super.equals(otherObject)) return false;
Manager.other = (Manager)otherObject;
// Usul super.equals tekshiradi
// this va otherObject obyektlari bitta sinfga tegishlimi.
return bonus == other.bonus;
} }
```

Object sinfining yana bir muxim usuli toString, bo‘lib obyektni satr shaklida qaytaradi. Bu usul deyarli hamma sinflarda qo‘shimcha yuklanadi, va obyekt xolatini bosmaga chiqarishga mo‘ljallangan.

Ko‘p (hammasi emas) toString usullari sinf nomi dan iborat bo‘lib, kvadrat qavslarda maydonlari qiymatlari ko‘rsatiladi. Quyida Employee sinfining toString usuli realizatsiyasi ko‘rsatilgan.

```
public String toString()
{
return "Employee[name" + name
+ ",salary =" + salary
+ ",hireDay =" + hireDay
}
```

Bu usulni takomillashtirish mumkin. Sinf nomini toString usuliga kiritmasdan, getClass().getName() usulini chaqiramiz va sinf nomini o‘z ichiga olgan satrni olamiz.

```
public String toString()
{
return getClass().geName()
+ "[name=" + name
+ ",salary=" + salary
+ ",hireday=" + hireDay
```

}

Endi toString usuli voris sinflar bilan ham ishlaydi.

Albatta voris sinf yaratgan lasturchi o‘z toString usulini yaratishi va voris sinf nomini qo‘shishi lozim. Agar supersinfda getClass ().getNamef() usuli chaqirilsa, voris sinf super.ToString ( ) usulini chaqiradi. Manager sinfida toString usuliga misol.

class manager extends Employee

{

public String toString()

{

return super.toString()+ "[bonus=" + bonus

Endi Manager sinfi obyekti xolati quyidagi shaklda chiqariladi:

Manager[name=...,salary=...,hireDay=...][bonus=...]

Agar obyekt satr bilan "+" amali yordamida konkatenatsiya qilinsa Java tili kompilyatori obyekt joriy xolatini olish uchun avtomatik ravishda toString usulini chaqiradi.

Birorr x — ixtiyoriy obyekt uchun dasturchi System.out.println(x) usulini chaqirsin;

Bu holda println usuli x. toString () usulini chaqiradi va natija satrini chiqaradi.

Object sinfida aniqlangan toString usuli sinf nomi va obyekt adresini chiqaradi. Masalan

System.out.println(System.out);

chaqirish natijasida quyidagi satr xosil bo‘ladi

java.io.PrintStream@2f668 4

Buning sababi shuki PrintStream sinfida toString usuli qo‘shimcha yuklanmagan.

Standart bibliotekaga tegishli ko‘p sinflarda toString usuli shunday aniqlanganki, uning yordamida dasturni sozlash uchun kerakli ma’lumot olish mumkin. Ba’zi sozlovchilar obyektlar xolatini ekranda akslantinrish uchun toString usulini

chaqirishga imkon beradi. Shuning uchun dastur trassirovkasida, quyidagi ifodalardan foydalanish mumkin

System.out.println{"Joriy xolat = " + position);

15.2. Umumlashgan dasturlash

Object tipidagi o‘zgaruvchilarda ixtiyoriy sinf o‘zgaruvchilari qiymati saqlanishi mumkin, masalan String sinfi:

Object obj = "Salom"; // To‘g‘ri.

Lekin sonlar, simvollar va mantiqiy o‘zgaruvchilar obyektlarga kirmaydi.

obj = 5 ; // Noto‘g‘ri.

obj = false; // Noto‘g‘ri.

Bundan tashqari hamma tipdagi massivlar, ularda obyektlar yoki asosiy tiplardagi o‘zgaruvchilar saqlanishiga qaramay Object sinfi vorisi hisoblanadi.

Employee staff [] = new Employee[10];

Object arr = staff; // To‘g‘ri.

arr = new int[10]; // To‘g‘ri.

Biror sinfga tegishli obyektlar massivini Object sinfi obyektlari massiviga aylantirish mumkin. Masalan, Employee[] sinfi massivini Object[ ] sinfi massivini kutayotgan usulga uzatish mumkin. Bu usul umumlashgan dasturlash uchun foydalidir (generic programming).

Quyida umumlashgan dasturlash konsepsiyasini ko‘rsatuvchi misol keltirilgan. Bu misolda massivda element indeksini aniqlash lozim.

```
static int find(Object[] a, Object key)
(int i;
For
 (i =0; i < a.length; i++)
if (a[i].equals(key)) return i;
return -1; // Indeks topilmagan.
}
```

Misol uchun,

Employee staff[] = new Employee[10];

Employee harry;

int n = find(staff, harry);

SHuni ta’kidlab o‘tish lozimki Object[ ] tpi massivini faqat biror sinf obyektlari massiviga aylantirish mumkin. Preobrazovat massiv tipa int[ ] v massiv tipa Object[] tipi massiviga int[ ] tipi massivini o‘zgartirish mumkin emas.

Agar biror sinf obyektlaridan massiv, Object[ ] tipidagi massivga aylantirilsa, umumlashgan massiv boshlang‘ich tip xaqidagi ma’lumotni o‘zida saqlab qoladi. Bu massivga boshqa tipdagi elementni joylashtirish mumkin emas.

15.3. Interfeys ta’rifi. Operator interface. Operator implements

Interfeys — bu usullar jamlamasi oshkor spetsifikatsiyasi bo‘lib, shu spetsifikatsiyani realizatsiya qilayotgan sinfda bu usullar ta’rifi albatta berilishi lozim. Interfeysda bu usullar realizatsiyasi berilmaydi. Abstrakt sinflar kabi interfeyslar ko‘plik vorislikda foydalanishi mumkin. Konkret sinf faqat bitta supersinf vorisi bo‘lishi mumkin lekin cheklanmagan sondagi interfeyslar realizatsiya qilinishi mumkin.

Java tilida interfeyslar usullarni dastur bajarilishi davomida dinamik qo‘llashni (resolution) amalga oshirish uchun ishlatiladi. Interfeyslar sinflarga o‘xshaydi, lekin usullar realizatsiyasi bo‘lmaydi. Sinf ixtiyoriy sondagi interfeysga ega bo‘lishi mumkin. Sinfda interfeylar usulari to‘la to‘plamini realizatsiya qilish lozim. Sinf bu usullari signaturasi shu sinf realizatsiya qilayotgan iterfeys usullari signaturasi bilan bir xil bo‘lishi lozim. Interfeyslar sinflar vorislikka asoslangan iyerarxiyasi bilan kesishmaydigan o‘z iyerarxiyasiga ega. Bu vorislik iyerarxiyasi bilan bog‘lanmagan turli sinflarda bitta intrfeysni realizatsiya qilishga imkon beradi. Interfeyslar kuchi shundan iborat. Interfeyslar C++ tilida ko‘plik vorislik mexanizmi analogidir, lekin ulardan foydalanish qulayroq

Interfeys ta’rifi sinf ta’rifiga o‘xshash, farqi shundaki interfeysda ma’lumotlar e’loni va konstruktorlar yo‘qdir. Interfeys ta’rifi umumiy ko‘rinishi:

interface nom { natija_tipi usul_nomi1(parametrlar ro‘yxati);

tip final1_o‘zgaruvchi_ismi = qiymat; }

Interfeysda e’lon qilingan usullar tanasi yo‘qdir. Interfeysda e’lon qilingan o‘zgaruvchilar ko‘zda tutilgan bo‘yicha final – o‘zgaruvchilar hisoblanadi. Shuning uchun realizatsiya qiluvchi sinfda ularning qiymatini o‘zgartirish mumkin emas. Bundan tashqari o‘zgaruvchilarni interfeysda ta’riflanganda konstanta qiymat bilan initsializatsiya qilinishi kerak. Quyida callback nomli usulga ega va int tipidagi parametrga ega interfeys ta’rifi berilgan.

interface Callback { void callback(int param); }

Operator implements — biror interfeys yoki interfeyslarni realizatsiya qiluvchi sinf ta’rifiga qo‘shimchadir

class sinf_nomi [extends supersinf]

[implements interfeys0 [, interfeys1...]] { sinf tanasi }

Agar sinfda bir nechta interfeyslar realizatsiya qilinsa, ularning nomlari vergul bilan ajratiladi. Quyida interfeysni realizatsiya qiluvchi sinfga misol berilgan:

class Client implements Callback {

void callback(int p) { System.out.println("callback called with " + p);

} }

Quyidagi misolda odin ta’rifi berilgan interfeys callback usuli, interfeysga ilova – o‘zgaruvchi orqali chaqiriladi:

class TestIface { public static void main(String args[]) { Callback s = new client();

c.callback(42); } }

Quyida dastur bajarilishi natijasi beilgan:

S:\> Java TestIface

callback called with 42

15.4. Interfeyslarda o‘zgaruvchilar

Interfeyslardan turli sinflarga birgalikda foydalaniluvchi konstantalarni import qilish uchun foydalanish mumkin. Bu holda biror sinfda interfeys realizatsiya qilinsa interfeys o‘zgaruvchilari nomlari bu sinfda konstanta sifatida ko‘rinadi. Bu S va C++

tillarida konstantlarni #define direktivasi yoki Pascal / Delphi tillarida const kalit so‘zi yordamida berishga mosdir.

Agar interfeys o‘z ichiga usullarni olmasa, interfeys realizatsiyasi deb e’lon qilingan sinf xech narsa realizatsiya qilmaydi. Konstantalarni sinf nomlar fazosiga import qilish uchun final modifikatorli o‘zgaruvchilardan foydalanish qulaydir.

```
import java.util.Random;
interface SharedConstants { int NO = 0;
int YES = 1;
int MAYBE = 2;
int LATER = 3;
int SOON = 4;
int NEVER = 5; }
class Question implements SharedConstants {
Random rand = new Random();
int ask() {
int prob = (int) (100 * rand.nextDouble());
if (prob < 30) return NO; // 30%
else if (prob < 60) return YES; // 30%
else if (prob < 75) return LATER; // 15%
else if (prob < 98) return SOON; // 13%
else return NEVER; // 2% } }
class AskMe implements SharedConstants {
static void answer(int result) {
switch(result) {
case NO:
System.out.println("No");
break;
case YES:
System.out.println("Yes");
```

```
break;
case MAYBE:
System.out.println("Maybe");
break;
case LATER:
System.out.println("Later");
break;
case SOON:
System.out.priniln("Soon");
break;
case NEVER:
System.out.println("Never");
break;
} }
public static void main(String args[]) {
Question q = new Question();
answer(q.ask());
answer(q.ask());
answer(q.askO);
answer(q.ask());
} }
```

E’tibor berinki dastur xar gal ishlatilganda xar xil natija beradi, chunki unda java.util paketiga tegishli Random tasodifiy sonlar generator ishlatilgan.

```
S:\> Java AskMe
Later
Scon
No
Yes
```

Nazorat uchun savollar

1. Interfeys ta’rifini keltiring.
2. Interfeys sinfdan qanday farq qiladi?
3. Nima uchun operator implements ishlatiladi?
4. Interfeys nima uchun ishlatiladi.
5. Birgalikda ishlatiladigan konstantalarni turli sinflarga import qilish uchun interfeysdan qanday foydalaniladi?

XULOSA

Ma’lumki, dastur tuzish sermashaqqat jarayon, lekin Delphi tizimi bu ishni sezilarli darajada soddalashtiradi va masala turiga qarab dastur tuzuvchi ishining 50–80%ni tizimga yuklaydi. Delphi tizimi dasturni loyihalash va yaratish vaqtini kamaytiradi, hamda Windows muhitida ishlovchi dastur ilovalarini tuzish jarayonini osonlashtiradi.

Delphi - Windows muhitida ishlaydigan dastur tuzish uchun qulay bo‘lgan vosita bo‘lib, kompyuterda dastur yaratish ishlarini avto- matlashtiradi, xatoliklarni kamaytiradi va dastur tuzuvchi mehnatini yengillashtiradi. Delphida dastur zamonaviy vizual loyihalash texnolo- giyasi asosida obyektga mo‘ljallangan dasturlash nazariyasini hisobga olgan holda tuziladi. Delphi tizimi Turbo Pascal 7.0. tilining rivoji bo‘lgan obyektga mo‘ljallangan Object Pascal dasturlash tilini ishlatadi.

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.

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.

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.

Funksiya shablonlari (parametrlangan turlar) bog‘langan funksiyalar oilasini tuzish imkonini beradi. Shablon kiritilishi uchun, hosil qilingan funksiya avtomatlashtirish, har xil tipli ma’lumotlarni qayta ishlashdan iborat. Masalan, algoritm tartiblash uchun har qaysi funksiyani o‘zini aniqlovchi tipi qo‘llaniladi.

C++ o‘zi istisno xolatlarni yuzaga keltirmaydi. Ularni C++ ning throw operatoridan foydalangan dasturlar yuzaga keltiradi. Istisno yuzaga kelganda, throw operatoridagi <ifoda> nom berish ifodasi muvaqqat obyektni nomlaydi (initsiallashtiradi), Bunda muvaqqat obyektning turi ifoda argumenti (dalili) ning turiga mos keladi. Ushbu obyektning boshqa nusxlari, masalan, istisno obyektidan nusxa ko‘chirish konstruktori yordamida generatsiya qilinishi mumkin.

GLOSSARIY

Atamaning o‘zbek tilida nomlanishi	Atamaning ingliz tilida nomlanishi	Atamaning rus tilida nomlanishi	Atamaning ma’nosi
Eksplikatsiya	**Explicatsion**	**Eksplikatsiya**	grafik obrazning ma’nosini ochib beruvchi ma’lumotlar to‘plami.
Zonal to‘r	**Net Zone**	**Zonalnaya setka**	maydonni qismlarga bo‘lib, har bir qismiga maxsus nom berishdan iborat.
Jadval to‘ri	**Grid table**	**Setka tablitsi**	bu o‘zaro kesishuvchi zonalar kombinatsiyasidan iborat.
Dasturiy interfeys	**software interface**	**Programmniy interfeys**	hisoblash mashinasi doirasida dasturlar va qurilmalarni o‘zaro muvofiq tarzda ishlashini ta’minlab beruvchi vositalar turkumi.
Foydalanuvchi interfeys	**Interface**	**Interfeys polzovatelya**	bu EXM yoki dasturlar ta’minoti bilan foydalanuvchi o‘rtasida muloqotni tashkil etuvchi dasturlar va apparatlar majmuasi.
Multimedia texnologiyasi	**Multimedia texnologies**	**Multimediynaya texnologiya**	foydalanuvchini kompyuter bilan muloqoti uchun ovoz, video, grafika, matn, animatsiya va boshqa vositalar yordamida tabiiy muhitni ta’minlash jarayoni.
Drawing	**Drawing**	**Drawing**	Rasm solish asboblar panelida joylashgan tugmalar tizimi.
Microsoft Clip Art	**Microsoft Clip Art**	**Microsoft Clip Art**	standart grafiklar va rasmlar bibliotekasi.
Word Art	**Word Art**	**Word Art**	Xar qanday tasvirlarni qo‘yish, bezash va matnlarni tahrirlash vositalari.
OLE	**OLE**	**OLE**	bir dasturdagi rasmlarni boshqa dasturdagi rasmlar bilan bog‘lovchi yoki kiritib qo‘yuvchi vositalar majmuasi.
Eguation Edition	**Eguation Edition**	**Eguation Edition**	matematik, fizik, mantiqiy simvollarni matnga kirituvchi amaliy dastur.
Microsoft Map	**Microsoft Map**	**Microsoft Map**	geografik kartalarni, regionlarni ularni elementlari va xarakteristikalari bilan birgalikda shakllantiruvchi amaliy dastur.
Fonli rasm	**wallpaper**	**Fonoviy risunok**	qog‘ozga to‘g‘ridan-to‘g‘ri tushirilgan va u bilan mustahkam bog‘langan rasm.
Chizilgan obyekt	**painted object**	**Narisovanniy obyekt**	qog‘ozda erkin siljib yuruvchi, lekin matn bilan to‘qnashmaydigan rasm.
Obyektni trans-formatsiyalash	**Transformatsiya obyekta**	**Transformatsiya obyekta**	obyektni o‘z o‘qi atrofida gorizontal va vertikal yo‘nalishda o‘zgartirish.
Diagramma	**Diagram**	**Diagramma**	bu Excel ishchi sahifasidagi ma’lumotlarni grafik tasvirlanishi.

Diagramma ustasi	**Chart Wizard**	**Master Diagrammi**	bu muloqot oynalar tizimi bo‘lib, u erda diagramma tuzish uchun kerakli barcha ishlar amalga oshiriladi.
Diagramma oblasti	**chart Area**	**Oblast diagrammi**	bu berilgan ma’lumotlar qatori bo‘lib, diagramma nomini va afsonalarni ham o‘z ichiga oladi.
Qurish oblasti	**Area byulding**	**Oblast postroeniya**	bu yuza diagrammalarda uning o‘klari bilan chegaralangan, xajmiy diagrammalarda esa kategoriyalar nomi, kesishuvchi belgilar va o‘klar nomini ham uz ichiga oladi.
Power Point	**Power Point**	**Power Point**	bu grafik dasturlar paketi bo‘lib, elektron slaydlarni tayyorlash, ular bilan tanishishni uyushtirish va slayd-filmlarni namoyish etishga tayyorlaydi.
Taqdimot	**Prezentation**	**Prezentatsiya**	bu slaydlar va maxsus effektlar to‘plami bo‘lib, ularni ekranda ko‘rsatish, tarqatiladigan material, dokladni plani va konspekti shaklida bitta faylda saqlanadi.
Slayd	**Slayd**	**Slayd**	bu prezentatsiyaning alohida kadri bo‘lib, o‘z ichiga matnni, sarlavhalarni, grafik va diagrammalarni olishi mumkin.
Tarkatiladigan material	**Handout**	**Razdatochniy material**	qulay shaklda bosib chiqarilgan va tanishish uchun mo‘ljallangan materiallar.
Dizayn qolipi	**design sketch**	**Eskiz dizayna**	professional tomonidan oldindan tayyorlab qo‘yilgan grafiklar, bo‘yoqlar, jilolar, tovushlar namunasi bo‘lib, ular slaydlarda ishlatish uchun mo‘ljallangan.
Animatsiya	**Animation**	**Animatsiya**	bu slaydlarni namoyish qilish va ko‘rsatishda ularni samaradorligini oshiruvchi tovush, rang, matn va harakatlanuvchi effektlar va ularni **yig‘indisidan iborat.**
Past sezuvchanlik	**highly sensitive**	**Nizkochuvstvitelniy**	200 vertikal nuqtalar x 320 gorizontal nuqtalar;
Yuqori sezuvchanlik:	**highly sensitive**	**Visokochuvstvitelniy**	a) 400 vertikal nuqtalar x 640 gorizontal nuqtalar; b) 800 vertikal nuqtalar x 640 gorizontal nuqtalar.
Below	**Below**	**Below**	tashqi ramkaning faqat pastki chizig‘i chizaladi
Above	**Above**	**Above**	tashqi ramkaning faqat yuqori chizig‘i chiziladi
Ale	**Ale**	**Ale**	barcha ichki ramkalar chiziladi.
ALIGN	**ALIGN**	**ALIGN**	atributi tasvirning nisbiy egallab turgan joyini boshqarish imkoniyatini yaratib beradi.
All files	**All files**	**All files**	sayt barcha fayllarining miqdori va umumiy o‘lchami.
ALT	**ALT**	**ALT**	atributi «alternativ matn» deb ataladigan grafik obraz paydo bo‘lishi lozim bo‘lgan matn satrini beradi
Box	**Box**	**Box**	tashqi ramkaning faqat hamma chiziqlari chiziladi.
BGCOLOR	**BGCOLOR**	**BGCOLOR**	jadval foni rangi.
BORDERCOLOR	**BORDERCOLOR**	**BORDERCOLOR**	ramka rangi.
CELLPADING	**CELLPADING**	**CELLPADING**	yacheyka ichidagi narsalar va chegarasi orasidagi masofa.
CELLPADING	**CELLPADING**	**CELLPADING**	yacheyka ichidagi narsalar bilan ramka orasidagi bo‘sh oraliq razmerini piksellarda beradi.
CELLSPACING	**CELLSPACING**	**CELLSPACING**	yacheyka orasidagi masofa.
CHECKBOX	**CHECKBOX**	**CHECKBOX**	Nazorat indikatori
Clarinet	**Clarinet**	**Clarinet**	foydalanish uchun ko‘pchilik servis markazlari bilan

			imzolanadigan katta yangiliklar xizmati.
Cols	**Cols**	**Cols**	faqat vertikal chiziqlar chiziladi (ustunlar orasidagi);
Comments	**Comments**	**Comments**	sharhlarni joylashtirish uchun.
DOMEN (DNS – DOMAIN NAME SYSTEM)	**DOMEN (DNS – DOMAIN NAME SYSTEM)**	**DOMEN (DNS – DOMAIN NAME SYSTEM)**	normalarning domen sistemasi; Internet tarmog‘idagi kompyuter nomlarini IP-adreslariga o‘tkazib beruvchi ma’lumotlar bazasining tarmoq sistemasi.
Elektron tarjimon	**The electronic interpreter**	**Elektronniy perevodchik**	o‘ziga yuborilgan matnni bir tildan ikkinchi tilga tarjima qilib beradi
Folders	**Folders**	**Folders**	sayt strukturasini aks ettirish rejimi.
FRAME	**FRAME**	**FRAME**	jadval ramkasining qaysidir tashqi qismi chizmasini aniqlaydi.
FTP (Fili Transfer Protocol)	**FTP (Fili Transfer Protocol)**	**FTP (Fili Transfer Protocol)**	fayllarni uzatuv protokoli; kompyuterlararo axborot almashuvining standart usuli.
Faks-servis	**Faks-servis**	**Faks-servis**	tarmoq faks serviridan foydalanib, foydalanuvchiga faksimal aloqa orqali xabarlar jo‘natish imkonini beradi
Gippermatn xujjat	**Gippermatn xujjat**	**Gippermatn xujjat**	bu boshqa xujjatlarga o‘tish uchun aloqa bog‘lovchi (sso‘lka)ni o‘zida saqlaydigan xujjat.
Gopher	**Gopher**	**Gopher**	Internet zaxira va imkoniyatlarni qidirish, ularga bog‘lanish va ulardan foydalanish uchun mo‘ljallangan interaktiv obolochka (qobig‘) foydalanuvchi bilan interfeys menyu sistemasi orqali olib boriladi.
HEIGHN	**HEIGHN**	**HEIGHN**	jadval balandligi
Hsides	**Hsides**	**Hsides**	tashqi ramkasining faqat gorizontal chizig‘i chiziladi, ya’ni yuqori va quyi chiziqlari
HTML (Hypertext Markyp Languge)	**HTML (Hypertext Markyp Languge)**	**HTML (Hypertext Markyp Languge)**	gippermatn xujjatlarini yozish uchun mo‘ljallangan til.
HTTP (Hyper Text Transfer Protocol)	**HTTP (Hyper Text Transfer Protocol)**	**HTTP (Hyper Text Transfer Protocol)**	bu Internet protokoli hisoblanib uning yordamida bir formatdagi ikki kompyuter o‘zaro bog‘lanib muloqot olib borish imkoniyatiga ega bo‘ladi.
Hyperlinks	**Hyperlinks**	**Hyperlinks**	ichki va tashqi aloqalar strukturasi rejimi.
Internet	**Internet**	**Internet**	Jaxondagi xar xil kompyuter tarmoqlari bilan aloqa bog‘lab turishni ta’minlovchi texnik vositalar, programma ta’minoti, standart va kelishuvlar yig‘indisi.
IP (Internet Protocol)	**IP (Internet Protocol)**	**IP (Internet Protocol)**	tarmoqdagi paketlarni marshrutlashni ta’minlovchi tarmoqlararo o‘zaro xarakat protokoli.
LAN (local area NetWork)	**LAN (local area NetWork)**	**LAN (local area NetWork)**	geografik bir joydagi lokal tarmoq.
Lhs	**Lhs**	**Lhs**	tashqi ramkaning faqat chap chizig‘i chiziladi
Linked files	**Linked files**	**Linked files**	sayt asosiy sahifasi bilan bog‘langan fayllar miqdori.
Microsoft Front Page	**Microsoft Front Page**	**Microsoft Front Page**	Web-uzelni dasturlashsiz tezda etkazish va unda professional ravishda tayyorlangan xujjatlarni nashr etish imoniyatini beradigan maxsus vosita hisoblanadi.
Modified Date	**Modified Date**	**Modified Date**	soni va qaysidir saytning oxirgi o‘zgarish vaqtini ko‘rsatadi.
Marshrutizator –	**Marshrutizator –**	**Marshrutizator –**	tarmoq paketlarini marshrutlash bilan shug‘ullanadigan

(roater)	**(roater)**	**(roater)**	kompyuter tarmog‘i, ya’ni paketlarning tarmoq bo‘ylab eng qisqa xarakat marshrutlarini tanlab beriladi.
NAME	**NAME**	**NAME**	qatorida fayl yoki papka nomi ko‘rsatiladi.
Navigation	**Navigation**	**Navigation**	sayt navigatsiyasi rejimi.
Netscape Communication	**Netscape Communication**	**Netscape Communication**	bu dunyodagi eng ommabop va eng ko‘p ishlatiladigan brauzer hisoblanadi.
NNTP (Net News Transfor Protocol)	**NNTP (Net News Transfor Protocol)**	**NNTP (Net News Transfor Protocol)**	tarmoq yangiliklarini uzatuvchi protokol.
NOC	**NOC**	**NOC**	Internet tarmoqlari orasida paydo bo‘ladigan xar xil muammolarni xal qiluvchi Internet xar bir tarmog‘ini xususiy ekspluatatsion markazi.
None	**None**	**None**	hech qanday ichki ramkalar bo‘lmaydi;
NSFNET	**NSFNET**	**NSFNET**	IP- texnologiyasida tashkil qilingan milliy ilmiy fondning xususiy tarmog‘i.
Page	**Page**	**Page**	loyixalash rejimi.
PAP (Password authentication protocol)	**PAP (Password authentication protocol)**	**PAP (Password authentication protocol)**	Cerverga ulovchi parollar sistemasi.
Pictures	**Pictures**	**Pictures**	sayt grafik fayllarining miqdori va umumiy hajmi.
Pictures	**Pictures**	**Pictures**	sayt grafik fayllarining miqdori va umumiy hajmi.
PPP (Post office protocol)	**PPP (Post office protocol)**	**PPP (Post office protocol)**	oddiy modem liniyalarini internetga kirishda ishlatiladigan kanal darajasidagi protokol(Analog Slip).
Protokol	**Protokol**	**Protokol**	ikki va undan ortiq mustaqil qurilma yoki protsessorlar o‘rtasida forma va protseduralarga reklama qiluvchi qoida va kelishuvlar yig‘indisi.
RADIO	**RADIO**	**RADIO**	Selektor tugmasi
Reports	**Reports**	**Reports**	sayt to‘g‘risidagi zaruriy axborotlarni aks ettirish rejimi.
RESET	**RESET**	**RESET**	Anketani tozalash tugmasi.
Resurs	**Resurs**	**Resurs**	Foydalanuvchi ixtiyoriga berilish imkoniyati bor bo‘lgan sistemaning mantiqiy yoki fizikaviy qismi.
Rows	**Rows**	**Rows**	faqat gorizontal chiziqlar chiziladi (satrlar orasidagi);
RULES	**RULES**	**RULES**	jadval ramkasini ichki qismining qanday chizilishini ko‘rsatadi.
Server – kompyuter	**The computer a server**	**Kompyuter server**	boshqalarga o‘z xizmatini tavsiya qiluvchi tarmoq kompyuteri, ya’ni foydalanuvchilarning talablari (savollari) bilan shug‘ullanadi.
Server – programma	**Program server**	**Programmniy server**	bitta kompyuter xizmatini boshqa kompyuterga taqdim etish imkonini yaratuvchi tarmoq kompyuter dasturi.
Servis markazi	**Service the centre**	**Servis tsentr**	Internetga ulangan ko‘plab kompyuter sistemalarini quvvatlovchi markaz.
Shlyuz	**Shlyuz**	**Shlyuz**	abonentga TCP/IP protokollari bilan ishlamaydigan tarmoqda xabarlarni jo‘natish imkonini beradi
Size lu Type	**Size lu Type**	**Size lu Type**	fayl o‘lchami va uning kengayishini ko‘rsatadi.
SLIP (Serial Line	**SLIP (Serial Line**	**SLIP (Serial Line**	oddiy modem liniyalarini Internetga kirishda ishlatiladigan

Internet Protocol)	**Internet Protocol)**	**Internet Protocol)**	jaxon darajasidagi protokol.
Slow pages	**Slow pages**	**Slow pages**	30 sekundan ortiq yuklanadigan HTML – fayllar miqdori («sust sahifalar
Status satri	**Status satri**	**Status satri**	foydalanuvchiga dastur tomonidan berilishi lozim bo‘lgan barcha xabarlar bu satrda xabar ko‘rinishida paydo bo‘ladi.
SUBMIT	**SUBMIT**	**SUBMIT**	Anketani jo‘natish tugmasi
Task	**Task**	**Task**	topshiriq va masalalarni boshqarish rejimi.
TCP (Transmission Control Protocol)	**TCP (Transmission Control Protocol)**	**TCP (Transmission Control Protocol)**	tarmoqdagi axborot uzatuvini nazorat qilib turuvchi protokol; katta xajimdagi axborotlarning jo‘natish muammolarini xal qiladi.
Telnet	**Telnet**	**Telnet**	uzoqda turib tarmoqdagi istagan kompyuterni boshqarish rejimi.
TEXT	**TEXT**	**TEXT**	Matnli maydon
Title	**Title**	**Title**	sahifa sarlavhasi yoki sayt qolgan elementlari nomini ko‘rsatadi.
Unlinked files	**Unlinked files**	**Unlinked files**	saytning asosiy sahifasi bilan to‘g‘ridan-to‘g‘ri bog‘lanmagan fayllar miqdori.
Usenet (Usenet Wewsq roupe)	**Usenet (Usenet Wewsq roupe)**	**Usenet (Usenet Wewsq roupe)**	tarmoq yangiliklari va tarmoqdagi elektron elonlar doskasini olish.
UUCP	**UUCP**	**UUCP**	bir Unix-xoctdan boshqasiga axborotlarni nusxalash protokoli. Ko‘plab pochta almashuv sistemalari shu protokolga asoslanib tuzilgan.
Uzel	**Uzel**	**Uzel**	tarmoqning asosiy vazifalarini bajaruvchi tarmoq kompyuteri.
Veronica (Very Easy Rodent - Oriented Vetwide Index to Computer Archives)	**Veronica (Very Easy Rodent - Oriented Vetwide Index to Computer Archives)**	**Veronica (Very Easy Rodent - Oriented Vetwide Index to Computer Archives)**	kalit so‘zlar bo‘yicha Internet tarmog‘ining ommaviy arxivida axborotlarni qidirish sistemasi.
Void	**Void**	**Void**	tashqi ramkani butunlay yo‘qotadi.
Vsides	**Vsides**	**Vsides**	tashqi ramkasining faqat vertikal chizig‘i chiziladi
WAIS (Wide Arle Information Service)	**WAIS (Wide Arle Information Service)**	**WAIS (Wide Arle Information Service)**	kalit so‘zlar bo‘yicha Internet tarmog‘ining ma’lumotlar bazasida kuchli axborotlar qidiruv sistemasi.
WAN (wide Area NetWork)	**WAN (wide Area NetWork)**	**WAN (wide Area NetWork)**	katta xududda joylashgan global tarmoq.
Whois	**Whois**	**Whois**	Internet tarmog‘ining adres kitobi.
WWW (World Wide Web)	**WWW (World Wide Web)**	**WWW (World Wide Web)**	xujjatlararo gipermatn aloqa bog‘lash qobiliyatiga ega bo‘lgan tarqoq ma’lumotlar bazasi sistemasi.
Xost	**Xost**	**Xost**	tarmoq vazifalaridan tashqari foydalanuvchilarning topshiriqlarini (programmalar, qisoblash ishlari va b.q.) bajaruvchi tarmoqning ishchi mashinasi ya’ni bosh EHM.

FOYDALANILGAN ADABIYoTLAR RO‘YXATI

I. O‘zbekiston Respublikasi qonunlari, Prezident farmonlari va qarorlari, Vazirlar mahkamasining qarorlari

1. O‘zbekiston Respublikasining Konstitutsiyasi. – T.: O‘zbekiston, 2017.- 46 b.

2. O‘zbekiston Respublikasi Prezidentining “O‘zbekiston Respublikasini yanada rivojlantirish bo‘yicha harakatlar strategiyasi to‘g‘risida» gi №PF-4947 sonli Farmoni. //Xalq so‘zi, 2017- yil 8- fevral

3. O‘zbekiston Respublikasi Prezidentining «2017-2021 yillarda O‘zbekiston Respublikasini rivojlantirishning beshta ustuvor yo‘nalishi bo‘yicha Harakatlar stratiegiyasini amalga oshirishga doir tashkiliy chora-tadbirlar to‘g‘risida»gi 2017- yil 14- fevraldagi F–4849-sonli Farmoyishi.

II. O‘zbekiston Respublikasi Prezidenti asarlari

4. Mirziyoyev SH.M. Buyuk kelajagimizni mard va olijanob xalqimiz bilan birga quramiz. – T.: “O‘zbekitson”, 2017.

5. Mirziyoyev SH.M. Konstitutsiya – erkin va farovon hayotimiz, mamlakatimizni yanada taraqqiy ettirishning mustahkam poydevoridir //Xalq so‘zi, 2017- yil, 8- dekabr.

6. Mirziyoyev SH.M. Qonun ustuvorligi va inson manfaatlarini ta’mindash – yurt taraqqiyoti va xalq farovonligining garovi. www.press-servic.uz.

7. Mirziyoyev SH.M. Millatlararo do‘stlik va hamjihatlik – xalqimiz tinchligi va farovonligining hayotbaxsh manbai//Milliy taraqqiyot yo‘limizni qat’iyat bilan davom ettirib, yangi bosqichga ko‘taramiz. – T: “O‘zbekiston” NMIU, 2017.

8. Mirziyoyev SH.M. Tanqidiy tahlil, qat’iy tartib-intizom va shaxsiy javobgarlik - har bir rahbar faoliyatining kundalik qoidasi bo‘lishi kerak. -T.: O‘zbekiston, 2017 y.,104 b.

9. O‘zbekiston Respublikasi Prezidentining «2017 – 2021 yillarda O‘zbekiston Respublikasini rivojlantirishning beshta ustuvor yo‘nalishi bo‘yicha Harakatlar strategiyasini «Faol tadbirkorlik, innovatsion g‘oyalar va texnologiyalarni qo‘llab-

quvvatlash yili»da amalga oshirishga oid davlat Dasturi to‘g‘risida» 2018- yil 22- yanvardagi PF-5308-sonli Farmoni.

III. Sohaga oid me’yoriy xujjatlar

10. O‘zbekiston Respublikasi Prezidentining «Axborot texnologiyalari soxasida kadrlar tayyorlash tizimini takomillashtirish to‘g‘risida» gi qarori, //Xalq so‘zi, 2005.

11. «Axborot-kommunikatsiya texnologiyalarini yanada rivojlantirishga oid qo‘shimcha chora-tadbirlar to‘g‘risida» O‘zbekiston Respublikasi Prezidentining 2005- yil 8- iyuldagi -117-sonqarori.

12. «ZiyoNET axborot tarmog‘ini yanada rivojlantirish to‘g‘risida» O‘zbekiston Respublikasi Vazirlar Mahkamasining 2005- yil 28-dekabrdagi 282-sonqarori.

IV. Asosiy adabiyotlar

13. Qobulov R.V. Obyektga yo‘naltirilgan dasturlash tillari: O‘quv qo‘llanma. –T.: TATU, 2013 y – 157 b.

14. Грехем И. Объектно ориентированные методы. Принципы и практика. Вильямс. 2004 г. – 879 стр.

15. Иванова Г.С. Объектно-ориентированое программирование. Учебник. МГТУ им Баумана, 2003 г. - 320 стр.

16. Лаптев, Валерий Викторович. С++ объектно-ориентированное программирование, учеб. Пособие / В.В. Лаптев.-М.;СПб: Питер, 2008 г.

17. Павловская Т.А. С++ Объектно-ориентированное программирование. Практикум. уч. пособие. – М.: СПб,; Нижний Новгород: Питер, 2006 – 265 стр.

18.Шмидский Я.К. Программирование на языке С++: Самоучитель. Учебное пособие. Диалектика. 2004 г. – 361 стр.

V. Qo‘shimcha adabiyotlar

19. Ашарина Н.А. Основы программирования на языке Си,.С++. Учебный курс. М.: 2012 г. – 215 стр.

20. Крупник Л.Б. Изучаем С++. Питер, 2003 г. – 251 стр.

21. Лишнер, Рей. С++. Справочник. М.-; СПб. Нижний Новгород: Питер, 2005 – 325 стр.

22. Мейерс С. Наиболее эффективное использование C++. 35 новых рекомендаций. ДИК-Пресс, 2008 г. – 304 стр.

23. Николенко Д.В. Самоучитель по Visual C++/СПб, 2001-245 стр.

VI. Davriy nashrlar, statistik to‘plamlar va hisobotlar

24. O‘zbekiston Respublikasi Mehnat va aholini ijtimoiy muhofaza qilish vazirligi ma’lumotlari

25. O‘zbekiston Respublikasi statistika qo‘mitasi ma’lumotlari

26. “Xalq so‘zi” gazetasining 2016-2018 yillardagi sonlari

27. “Iqtisodiyot va ta’lim” jurnalining 2016-2018 yillardagi soni.

VII. Internet saytlari

28. www.gov.uz – O‘zbekiston xukumati portali.

29. www.edu.uz – O‘zbekiston respublikasi oliy va o‘rta maxsus ta’lim vazirligi sayta.

30. www.ictcouncil.gov.uz- Kompyuterlashtirishni rivojlantirish buyicha Vazirlar Maxkamasi muvoffiqlashtiruvchi Kengashining sayti.

31. ziyonet.uz – Ziyonet internet tarmog‘i.

32. www.lex.uz – O‘zbekiston qonun hujjatlari sayti.

O‘.T. Xayitmatov, R.X. Alimov, A.A. Akramov,
O.X. Azamatov

OBYEKTGA YO‘NALTIRILGAN DASTURLASH TILLARI

O‘quv qo‘llanma

“IQTISODIYOT” – 2019.

Muharrir
Mirhidoyatova D.

Musahhih
Matxo‘jayev A.O.

Litsenziya AI № 240 04.07.2013. Terishga berildi 10.09.19. Bosishga ruxsat etildi 10.09.2019. Qog‘oz bichimi 60x80 1/16. Times garniturasi. Ofset bosma. Ofset qog‘ozi. Shartli bosma tabog‘i 8,7. Hisob nashr varag‘i 8,4. Adadi ___ nusxa.

“IQTISODIYOT” nashriyoti DUKning matbaa bo‘limida chop etildi.
100003. Toshkent shahri Islom Karimov ko‘chasi, 49-uy.

000000 Obyektga yo‘naltirilgan dasturlash tillari. O‘quv qo‘llanma. /Xayitmatov O‘.T., Alimov R.X., Akramov A.A., Azamatov O.X. - T.: «IQTISODIYOT», 2019. – 139 bet.

ISBN 00000000

UO‘K 330.115
KBK 000000