**№2-LABORATORIYA ISHI**

**2.1. Sinflarni qo‘llash. Sinflarning konstruktor va destruktorlari.**

* 1. **Sinflarni qo‘llash. Do‘stona va virtual funksiyalar**
  2. **STL komponentalar va konteynerlar bilan tanishish.**
  3. **Foydalanuvchining shablonlari.**

**Ishdan maqsad:** Talabalar bilan sinflarni qo‘llash, sinflarning konstruktor va destruktorlari hamda do‘stona va virtual funksiyalar ustida amaliy mashqlar bajarish.

**Qo‘yilgan masala:** Individual berilgan masalalar uchun sinflarni qo‘llash orqali dasturiy ta’minot yaratish va STL komponentalari va konteynerlar orqali dastur bajariladi.

**Ish bajarish tartibi:**

* Tajriba ishi nazariy ma’lumotlarini o‘rganish;
* Berilgan topshiriqning algoritmini ishlab chiqish;
* Python dasturlash muhitida dasturni yaratish;
* Natijalarni tekshirish;
* Hisobotni tayyorlash va topshirish.

**2.1. Sinflarni qo‘llash. Sinflarning konstruktor va destruktorlari.**

Boshqa umumiy maqsadli tillar singari, python ham boshidan beri obyektga yo‘naltirilgan til hisoblanadi. *Python* - obyektga yo‘naltirilgan dasturlash tili. Bu bizga obyektga yo‘naltirilgan yondashuv yordamida dasturlarni ishlab chiqishga imkon beradi. Pythonda biz osongina sinflar va obyektlarni yaratishimiz va ulardan foydalanishimiz mumkin.

**Obyektga yo‘naltirilgan dasturlash tizimining asosiy printsiplari quyida keltirilgan:**

* Object (*Obyekt*);
* Class (*Sinf*);
* Method (*metod, usul*);
* Inheritance (*Meros olish*);
* Polymorphism (*Polimorfizm*);
* Data Abstraction *(Ma'lumotlarni olish);*
* Encapsulation *(Inkapsulyatsiya);*

*Object. Obyekt* - bu holat va xulq-atvor, xususiyatlarga ega bo‘lgan shaxs. Bu sichqoncha, klaviatura, stul, stol, ruchka va boshqa turdagi har qanday haqiqiy obyekt bo‘lishi mumkin. Python-dagi hamma narsa obyekti bo‘lib, deyarli hamma narsada atributlar va metodlar mavjud. Barcha funksiyalar funksiya manba kodida belgilangan *\_\_doc\_\_* qatorini qatorini qaytaradigan o‘rnatilgan doc atributiga ega.

*Class (Sinf).*Sinf obyektlar to‘plami sifatida aniqlanishi mumkin. Bu ba'zi bir o‘ziga xos atributlar va usullarga ega bo‘lgan mantiqiy shaxs. Masalan: agar sizda ishchilar sinfingiz bo‘lsa, unda u atribut va usulni, ya'ni elektron pochta identifikatori, ism, yosh, ish haqi va boshqalarni o‘z ichiga olishi kerak.

***Sintaksis:***

*class ClassName:*

*…*

*…*

*Method (metod, usul).**Metod* - bu obyekt bilan bog‘liq bo‘lgan funksiya. Python-da metod faqat sinf misollari uchun xos emas. Har qanday obyekt turi metodlariga ega bo‘lishi mumkin.

*Inheritance (Meros olish).* *Merosxo‘rlik* - bu haqiqiy dunyo meros tushunchasini simulyatsiya qiladigan obyektga yo‘naltirilgan dasturlashning eng muhim jihati. Bola obyekti ota-onaning barcha xususiyatlarini va xatti-harakatlarini egallashini belgilaydi. Merosdan foydalanib, biz boshqa sinfning barcha xususiyatlari va xatti-harakatlaridan foydalanadigan sinfni yaratishimiz mumkin. Yangi sinf hosil bo‘lgan sinf yoki bola klassi, xossalari olingan sinf esa asosiy sinf yoki ota-ona sinfi sifatida tanilgan. Bu kodning qayta ishlatilishini ta'minlaydi.

*Polymorphism (Polimorfizm).* *Polimorfizm* tarkibida ikkita "poli" va "morflar" so‘zlari mavjud. Poli ko‘p, morflar esa shakllar degan ma'noni anglatadi. Polimorfizm bilan biz bitta vazifani har xil usulda bajarish mumkinligini tushunamiz. Masalan, sizda sinf hayvonlari bor, va barcha hayvonlar gapirishadi. Ammo ular boshqacha gapirishadi. Bu erda "gapirish" harakati ma'noda polimorf va hayvonga bog‘liq. Shunday qilib, mavhum "hayvon" tushunchasi aslida "gapirmaydi", lekin aniq hayvonlar (it va mushuklar kabi) "gapirish" harakatini aniq amalga oshiradilar.

*Encapsulation (Inkapsulyatsiya).* *Inkapsulyatsiya* – obyektga yo‘naltirilgan dasturlashning muhim jihati hisoblanadi. U metodlar va o‘zgaruvchilarga kirishni cheklash uchun ishlatiladi. Inkapsulyatsiya kod va ma’lumotlar tasodifan o‘zgartirilishidan bir bir ichida birlashtiriladi.

*Data abstraction (Ma'lumotlarni abstraktsiya qilish).*Ma'lumotlarni ajralish va inkapsulyatsiya qilish ikkalasi ham ko‘pincha sinonim sifatida ishlatiladi. Ikkalasi ham deyarli sinonimdir, chunki ma'lumotlar abstraktsiyasiga inkapsulyatsiya orqali erishiladi. Abstraktsiya ichki tafsilotlarni yashirish va faqat funksionallikni ko‘rsatish uchun ishlatiladi. Biron bir narsani mavhumlashtirish, bu narsa funksiyalar yoki butun dastur bajaradigan ishlarning mohiyatini o‘z ichiga olishi uchun narsalarga nom berishni anglatadi.

*Pythonda sinf va obyektlar.*Biz allaqachon muhokama qilganimizdek, sinf virtual obyekt bo‘lib, uni obyektning rejasi sifatida ko‘rish mumkin. Sinf paydo bo‘lganida obyekt paydo bo‘ldi. Keling, buni bir misol orqali tushunaylik. Aytaylik, sinf binoning prototipidir. Bino polga, eshiklarga, derazalarga va hokazolarga oid barcha ma'lumotlarni o‘z ichiga oladi, biz ushbu detallarga asoslanib, xohlagancha bino yasay olamiz. Demak, binoni sinf sifatida ko‘rish mumkin va biz shu sinfning shuncha obyektini yaratishimiz mumkin. Boshqa tomondan, obyekt sinfning misoli. Obyektni yaratish jarayonini instantatsiya deb atash mumkin. O‘quv qo‘llanmasining ushbu qismida biz python-da sinflar va obyektlarni yaratishni muhokama qilamiz. Atributga sinf obyekti yordamida qanday erishish mumkinligi haqida ham gaplashamiz.

*Python* – obyektga yo‘naltirilgan dasturlash tili. Pythonda deyarli barcha narsa obyekt hisoblandi. Ularning o‘z xususiyatlari va funksiyalari bor. Sinflar esa obyekt konstruktorlari hisoblanadi. Ular bilan obyektlar tuziladi.

*Sinf hosil qilish.*Sinf hosil qilish uchun class kalit so‘zi ishlatiladi. Hozir biz Son degan sinf hosil qilamiz. Shu sinf nomini print so‘zi bilan ekranga chiqarish buyrug‘ini bersak, shu sinf mavjudligi haqida ma’lumot chiqadi:

*class Son:*

*x = 5*

*print(Son)*

*Obyekt hosil qilish.*Sinflar obyekt konstruktorlari ekanligini aytgan edik. Hozir yuqorida hosil qilgan sinfimiz orqali yangi obyekt hosil qilamiz. Uning nomi s1 bo‘ladi.

*class Son:*

*x = 5*

*s1 = Son()*

*print(s1.x)*

***Natija****: 5*

*\_\_init\_\_() funksiyasi.*Yuqoridagi misollarimizdagi sinf va obyektlar bilan shunchaki sodda ko‘rinishda tanishib chiqdik. Ammo ular haqiqiy dasturlar tuzishga yaroqsiz. Sinflarning mohiyatini tushunish uchun *\_\_init\_\_()* ichki funksiyasini bilishimiz lozim.

Har bir sinf tuzilgan paytda *\_\_init\_\_()* funksiyasi mavjud bo‘ladi. *\_\_init\_\_()* funksiyasi obyektlar tuzilayotgan paytda ularning xususiyatlariga qiymatlarni yoki bajarilishi kerak bo‘lgan operatsiyalarni biriktiradi.

Hozir Ishchi degan sinf hosil qilamiz va unda ism va yosh ko‘rsatkichlariga qiymatlar o‘zlashtirish uchun *\_\_init\_\_()* funksiyasidan foydalanamiz.

Keyin *\_\_init\_\_()* funksiyasi har safar yangi obyekt tuzilganda avtomatik tarzda ishlaydi.

Eslatib o‘tamiz, *\_\_init\_\_()* funksiyasini yozayotganda har ikkala tarafdan ham ikkitadan ( \_\_ ) tag chiziq yoziladi.

*class Ishchi:*

*def \_\_init\_\_(self, ism, yosh):*

*self.ism = ism*

*self.yosh = yosh*

*p1 = Ishchi ("Salim", 31)*

*print(p1.ism) print(p1.yosh)*

*Obyekt funksiyalari.*Obyektlar ham funksiyaga ega bo‘lishi mumkin. Bu funksiyalar sinf ichida tuziladi va obyektlar tomonida ishlatiladi. Masalan, obyekt o‘zini tanishtirish funksiyasini tuzamiz:

*class Ishchi:*

*def \_\_init\_\_(self, ism, yosh):*

*self.ism = ism*

*self.yosh = yosh*

*def tanish(self):*

*print("Mening ismim "+ self.ism)*

*p1 = Ishchi ("Salim ", 20)*

*p1.tanish()*

**Natija**: *Mening ismim Salim*

*Self parametri.* *self* parametri sinfga tegishli o‘zgaruvchilarga murojaat qila olish uchun ishlatiladi. U o‘ziga xos yo‘llovchi vositadir. U aynan self deb nomlanishi shart emas, boshqa nomlarni ishlatish ham mumkin. Faqat u sinfdagi istalgan funksiyaning ilk parametri sifatida yozilishi shart.

Hozir yuqoridagi misolimizdagi self parametrlarini abc deb o‘zgartiramiz va natija o‘zgarmaydi.

*class Ishchi:*

*def \_\_init\_\_(abc, ism, yosh):*

*abc.ism = ism*

*abc.yosh = yosh*

*def tanish(abc):*

*print("Mening ismim "+ abc.ism)*

*p1 = Ishchi ("Salim", 31)*

*p1.tanish()*

**Natija:**

Mening ismim Salim

*Obyekt xususiyatini o‘zgartirish.* Biror obyektning xususiyatlarini osongina o‘zgartirishimiz mumkin.

Masalan, dastlab tuzgan obyektimiz 31 yosh bo‘lsa, so‘ng uni 33 yoshga o‘zgartiramiz:

*class Ishchi:*

*def \_\_init\_\_(abc, ism, yosh):*

*abc.ism = ism*

*abc.yosh = yosh*

*p1 = Ishchi ("Salim", 31)*

*p1.yosh = 33*

*print(p1.yosh)*

**Natija**: *33*

*Obyekt xususiyatini o‘chirish.* Obyekt xususiyatlarini o‘chirish ham mumkin. Hozir obyektimizdagi yosh xususiyatini o‘chiramiz. So‘ng uni ekranga chiqarish buyrug‘ini beramiz. Dastur ishga tushgach xatolik haqida xabar beriladi.

*class Ishchi:*

*def \_\_init\_\_(abc, ism, yosh):*

*abc.ism = ism*

*abc.yosh = yosh*

*p1 = Ishchi ("Salim", 31)*

*del p1.yosh*

*print(p1.yosh)*

**Natija:**

AttributeError: 'Ishchi' object has no attribute 'yosh'

*Obyektni o‘chirish.*Obyektni o‘chirish uchun del kalit so‘zini obyekt nomi bilan qo‘llaymiz. Natijada obyekt butkul o‘chib ketadi. Quyidagi kodimizda ham xatolik haqida xabar beriladi. Sababi, biz o‘chib ketgan obyektni ekranga chiqarmoqchi bo‘lyabmiz:

class Ishchi:

def \_\_init\_\_(abc, ism, yosh):

abc.ism = ism

abc.yosh = yosh

p1 = Ishchi ("Salim", 31)

del p1

print(p1)

**Natija:**

NameError: name 'p1' is not defined

* 1. **Sinflarni qo‘llash. Do‘stona va virtual funksiyalar**

*Konstruktor* - bu sinfning instansiya a'zolarini initsializatsiya qilish uchun ishlatiladigan maxsus metod (funksiya) turi.

Konstruktorlar ikki xil bo‘lishi mumkin:

* Parametrlangan konstruktor
* Parametrlanmagan konstruktor

Ushbu sinf obyektini yaratganimizda konstruktor ta'rifi bajariladi. Shuningdek, konstruktorlar obyekt uchun biron bir ishga tushirish vazifasini bajarish uchun yetarli resurslar mavjudligini tasdiqlaydilar.

*Pythonda konstruktor yaratish.*Pythonda *\_\_init\_\_* metodi sinf konstruktorini simulyatsiya qiladi. Ushbu usul sinfni qo‘zg‘atganda chaqiriladi. Biz *\_\_init\_\_* ta'rifiga qarab, sinf obyektini yaratishda istalgan sonli argumentlarni berishimiz mumkin. Bu asosan sinf atributlarini ishga tushirish uchun ishlatiladi. Har bir sinf konstruktorga ega bo‘lishi kerak, hatto u oddiygina konstruktorga tayansa ham.

Employee sinfining atributlarini ishga tushirish uchun quyidagi misolni ko‘rib chiqing.

***Example***:

*class Employee:*

*def \_\_init\_\_(self,name,id):*

*self.id = id*

*self.name = name*

*def display (self):*

*print("ID: %d \nName: %s"%(self.id,self.name))*

*emp1 = Employee("Rustam",101)*

*emp2 = Employee("Elyor",102)*

*emp1.display()*

*emp2.display()*

***ID****: 101*

***Name****: Rustam*

***ID****: 102*

***Name*** *Elyor*

*Sinflarda Vorislik tushunchasi.**Vorislik* - bu atama sinflarga xosdir. Vorislik deb bir sinfdagi barcha funksiya va xususiyatlarni boshqa bir sinf o‘ziga o‘zlashtirishiga aytiladi. Funksiyalari meros qilib olinadigan sinf ona sinf deyiladi. Meros qilib olingan funksiyalarni o‘ziga o‘zlashtiradigan sinf voris sinf deyiladi.

*Ona sinf hosil qilish.*Istalgan sinf ona sinf bo‘lishi mumkin. Shu sababli ona sinfni hosil qilish xuddi oddiy sinfni hosil qilish kabidir. Hozir Odam degan sinf hosil qilamiz. Unda ism va familiya parametrlari va tanish degan funksiyasi bo‘ladi. So‘ngra shu sinf orqali ***x*** obyekt hosil qilamiz:

*class Odam:*

*def \_\_init\_\_(self, ism, familiya):*

*self.ism = ism*

*self.familiya = familiya*

*def tanish(self):*

*print(self.ism, self.familiya)*

*x = Odam ("Salim", "Karimov")*

*x.tanish()*

**Natija**: Salim Karimov

*Voris sinf hosil qilish*. Voris sinf hosil qilish uchun yangi sinf tuzilayotganda ona sinfni paramet sifatida kiritamiz. Shunda voris sinf ona sinfdan barcha xususiyatlarni o‘zlashtiradi. Hozir ***Talaba*** degan sinf hosil qilamiz. Odam sinfi uning onam sinfi bo‘ladi. Qavslar ichida ona sinfni kirittamiz va uning barcha xususiyatlarini voris sinf o‘zlashtiradi. Qo‘shimcha parametr qo‘shish shart emas, ammo sinf hosil qilayotganda ichi bo‘sh bo‘lishi ham mumkin emas. Agar hechnarsa yozishni istamasak xatolik yuz bermasligi uchun pass kalit so‘zini qo‘shib qo‘yamiz:

*class Odam:*

*def \_\_init\_\_(self, ism, familiya):*

*self.ism = ism self.familiya = familiya*

*def tanish(self):*

*print(self.ism, self.familiya)*

*#Endi voris sinf ya'ni bola sinfni hosil qilamiz*

*class Talaba (Odam):*

*pass x = Talaba ("Salim", "Karimov")*

*x.tanish()*

*\_\_init\_\_()* *funksiyasini qo‘shish.*Avvalgi misolimizda voris sinf hosil qilganimizda pass kalit so‘zi bilan cheklanib qo‘ya qoldik. Shu sababli voris sinf barcha funksiyalarni avtomatik tarzda o‘zlashtirgan edi. Endi voris sinfga *\_\_init\_\_()* funksiyasi bilan parametrlarini joylashtiramiz. Bunda voris sinf ona sinfdagi *\_\_init\_\_()* Funksiyasidan emas o‘zidagidan foydalanadi.

*class Odam:*

*def \_\_init\_\_(self, ism, familiya):*

*self.ism = ism*

*self.familiya = familiya*

*def tanish(self):*

*print(self.ism, self.familiya)*

***# Endi voris sinf ya'ni bola sinfni hosil qilamiz***

*class Talaba (Odam):*

*def \_\_init\_\_(self, ism, familiya):*

*self.ism = ism*

*self.familiya = familiya*

*x = Talaba ("Aliddin ", "Valiyev")*

*x.tanish()*

**Natija:**

Aliddin Valiyev

* 1. **STL komponentalar va konteynerlar bilan tanishish.**

***Standart shablon kutubxonasi*** *(STL - Standard Template Library)* – tashqi ko‘rinishi barcha professional dasturchilar tomonidan bir ovozdan kutib olingan dasturiy mahsulotlardan biridir.

Deyarli barcha zamonaviy C ++ kompilyatorlarida STL kutubxonasi mavjud. Dastlab, deyarli har qanday ma'lumotlar bilan ishlash uchun tuzilgan, bu funksiyalar va sinflar shablonlari to‘plami bilan ta'minlanadi.

STL tarkibini to‘rtta asosiy tarkibiy qismlar tashkil etadi:

* ***iterator* :** algoritmni konteyner tarkibiga kirish uchun algoritm bilan ta'minlaydi.
* ***algoritm* :** hisoblash tartibini belgilaydi.
* ***konteyner* :** xotiradagi obyektlar to‘plamini boshqaradi.
* ***funksiya obyekti* :** boshqa tarkibiy qismlar tomonidan foydalanish uchun obyektda funksiyani kapsüllaydi.
* **adapter :** boshqa interfeys bilan ta'minlash uchun komponentni moslashtiradi**.**

**Qaytaruvchilar:**

*Iterator* - bu konteyner elementlariga kirishni ta'minlaydigan va ularni saralashga imkon beradigan obyekt. Iterator - bu konteynerning mulki. C ++ standart kutubxonasining birinchi amaliyotlarida konteyner elementiga ko‘rsatgich sifatida iterator amalga oshirildi. Zamonaviy dasturlarda bu ko‘rsatgichni konteyner obyektiga o‘tkazadigan sinfdir.

Iteratorlarga qo‘yiladigan asosiy talablar - bu konferentsiya va o‘sish operatorlarining mavjudligi. Quyida iterator konteyneri uchun misol.

*template<typename TYPE>*

*struct a\_container {*

*struct an\_iterator {*

*void operator++(); TYPE& operator\*();*

*};*

*an\_iterator begin();*

*an\_iterator end(); };*

***Algoritmlar.*** Men algoritmni odatdagidan ko‘ra aniqroq aniqlay olmayman: *algoritm - bu cheklangan sonli bosqichlarda istalgan natijaga olib keladigan harakatlar ketma-ketligi* .

STL holatida algoritmlar kirish parametrlari sifatida iteratorlarning yarim intervallarini oladigan shablon funksiyalari orqali amalga oshiriladi. Ushbu algoritmlarning umumiy ma’nosi quyidagicha tavsiflanadi:

*template<typename ITERATOR, typename RESULT>*

*RESULT an\_algorithm(ITERATOR first, ITERATOR last, ...);*

STL kutubxonasida qidirish, konvertatsiya qilish, saralash, nusxalash va hk kabi ba'zi bir standart harakatlarni bajaradigan funksiyalar guruhi mavjud, ular algoritmlar deb ataladi. Algoritmlar uchun parametrlar, qoida tariqasida, iteratorlardir. Algoritm unga berilgan iterator turiga ahamiyat bermaydi. Asosiysi shundaki, ikkinchisi ma'lum bir toifaga kiradi. Masalan, agar algoritm parametri bir yo‘nalishli iterator bo‘lishi kerak bo‘lsa, u holda almashtiriladigan iterator bitta yo‘naltirilgan yoki ikki yo‘naltirilgan yoki tasodifiy kirish iteratori bo‘lishi kerak.

Algoritmning misoli tengdir. Kirish iteratorlari tomonidan yuborilgan ikkita ma'lumot zanjirini taqqoslaydi va quyidagicha tavsiflanadi:

*template <class InputIterator1, class InputIterator2>*

*bool equal(InputIterator1 first1, InputIterator1 last1, InputIterator2 first2);*

Birinchi parametr - taqqoslanadigan ma'lumotlarning birinchi zanjiriga ishora qiluvchi kirish iteratoridir. Ikkinchisi ma'lumotlar diapazoni ko‘rsatkichining oxiriga murojaat qiladi. Uchinchi parametr - taqqoslangan ma'lumotlarning ikkinchi zanjiri. Va bu erda v1 va v2 ikki vektorni (massivlarni) taqqoslashning bir qismi:

*bool isEqual = teng (v1.begin (), v1.end (), v2.begin ());*

Bu erda standart vektor usullari qo‘llaniladi: start () ma'lumotlar zanjiri boshiga o‘rnatilgan iteratorni qaytaradi va end () diapazondan tashqaridagi ko‘rsatkichni qaytaradi. Agar vektorlarning barcha elementlari bir-biriga teng ravishda juft bo‘lsa, unda tenglik "true" (true) qiymatini qaytaradi.

E'tibor bering, barcha algoritmlarni ikkita asosiy toifaga bo‘lish mumkin: ma'lumotlarni o‘zgartiradigan va ularni o‘zgartirmaydigan.

***Konteynerlar.*** STL kutubxonasida uning arsenalida konteyner deb nomlangan elementlar mavjud. Konteynerlar o‘z ichiga boshqa narsalarni saqlaydigan obyektlardir. STLda o‘nta shunday konteyner mavjud:

* ***vector*  -** tasodifiy kirishga ega bo‘lgan massiv, zanjirning oxiriga ketma-ket ma'lumotlarni qo‘shish kerak bo‘lganda ko‘pincha foydalaniladi;
* ***list*  -** vektorga o‘xshash, ammo zanjirning biron bir joyida ma'lumotlarni qo‘shish va o‘chirishda samarali;
* ***deque*  -** boshida yoki oxirida ma'lumotlarni kiritish uchun qulay bo‘lgan konteyner;
* ***set*  -** ma'lum bir tartibda tartiblangan noyob elementlar to‘plami;
* ***multiset*  -** to‘plam bilan bir xil, ammo ularning nusxalari bo‘lishi mumkin;
* ***map*  -** tugmachalar orqali qiymatlarga kirishni ta'minlaydi;
* ***multimap* -** xarita bilan bir xil, ammo takroriy kalitlarga ruxsat berish;
* ***stack* -** ma'lumotlar bitta tartibda qo‘shiladi va aksincha olinadi;
* ***queue*  -** ma'lumotlar qo‘shiladi va shu tartibda chiqariladi;
* ***priority queue* -** navbat bilan bir xil, ammo ma'lumotni ustuvorligi bo‘yicha saralashi mumkin.

Shuni ta'kidlash kerakki, barcha algoritmlar ularni amalga oshirish tafsilotlarini bilmasdan konteyner usullari bilan ishlaydi. Shunday qilib, agar algoritm konteynerdan bitta elementning ikkinchisiga tengligini aniqlashi kerak bo‘lsa, u shunchaki konteynerda amalga oshirilgan taqqoslash operatorini "operator = = ()" deb ataydi.

***Funktsional obyektlar.*** Funktsional obyektlar "operator () ()" deb nomlangan ortiqcha yuklangan operator qo‘ng‘iroq funksiyasiga ega bo‘lgan obyektlardir. Ular samarali foydalanish uchun juda muhimdir. Ko‘rsatkichlarni funksiyaga o‘tkazish kerak bo‘lgan joylarda, funksiyani chaqirish uchun ortiqcha yuklangan operator bilan obyektni qabul qiladigan interfeys yaratiladi. Odatda barcha operatorlar chiziq shaklida yoziladi, bu tezlikni qo‘shimcha ravishda oshirishga imkon beradi.

Funktsional obyektlar barchasi arifmetik operatorlardir: qo‘shish (qo‘shish), ayirish (qo‘shish), ko‘paytirish (marta), bo‘linish (bo‘linish), qolgan qismini (modul) olish va belgini teskari (negate). Tenglikni (teng\_to), tengsizlikni (teng bo‘lmagan\_to), operatsiyalar "ko‘proq" (katta), "kamroq" (kamroq), "ko‘proq yoki teng" (katta) . Mantiqiy operatorlar o‘zlarining funktsional obyektlariga ega: mantiqiy "va" (mantiqiy\_ va), mantiqiy "yoki" (mantiqiy\_or) va mantiqiy "emas" (mantiqiy\_not).

Borland C ++ Builder to‘plamining tarkibiy qismi bo‘lgan STL kutubxonasidan foydalanib dasturga misol keltirilgan . Bu kichkina telefon ma'lumotnomasi.

Shuni ta'kidlash kerakki, barcha kiritilgan fayllar .h va .hpp kengaytmalarisiz aytib o‘tiladi. Bu tez orada barcha C ++ kompilyatorlarida qo‘llanilishi mumkin bo‘lgan yangi fayl kiritish usulidir.

*// konteynerlar xaritasidan foydalanish*

*# include <map>*

*// asosiy tarkibiy qism STL faylini kiritish*

*# include <algorithm>*

*// konteyner vektoridan foydalanish*

*# include <vector>*

*# include <iostream.h>*

*# include <string>*

*using namespace std;*

*// Ikki xaritalar e'lon qilindi*

*typedef map<string, long, less<string> > friendMap;*

*typedef map<long, string, less<long> > sortedMap;*

***ma'lumot xaritalarida saqlanadigan turlar uchun nomlar yozing***

*typedef friendMap::value\_type entry\_type;*

*typedef sortedMap::value\_type sorted\_entry\_type;*

*void printEntry(const entry\_type & entry)*

*{ cout << entry.first << ":" << entry.second << endl; }*

*void printSortedEntry(const sorted\_entry\_type & entry)*

*{ cout << entry.first << ":" << entry.second << endl; }*

*int prefix(const entry\_type& entry)*

*{ return entry.second / 10000; }*

*bool prefixCompare(const entry\_type & a, const entry\_type & b)*

*{ return prefix(a) < prefix(b); }*

*class checkPrefix {*

*public:*

*checkPrefix (int p) : testPrefix(p) { }*

*int testPrefix;*

*bool operator () (const entry\_type& entry)*

*{ return prefix(entry) == testPrefix; }*

*};*

*class telephoneDirectory {*

*public:*

*void addEntry (string name, long number)*

*{ database[name] = number; }*

*void remove (string name)*

*{ database.erase(name); }*

*void update (string name, long number)*

*{ remove(name); addEntry(name, number); }*

*void displayDatabase()*

*{ for\_each(database.begin(), database.end(), printEntry); }*

*void displayPrefix(int);*

*void displayByPrefix();*

*private:*

*friendMap database;*

*};*

*void telephoneDirectory::displayPrefix(int prefix)*

*{*

*cout << "Listing for prefix " << prefix << endl;*

*map<string, long, less<string> >::iterator where;*

*where = find\_if(database.begin(), database.end(), checkPrefix(prefix));*

*while (where != database.end()) {*

*printEntry(\*where);*

*where = find\_if(++where, database.end(), checkPrefix(prefix));*

*}*

*cout << "end of prefix listing" << endl;*

*}*

*void telephoneDirectory::displayByPrefix()*

*{*

*cout << "Display by prefix" << endl;*

*sortedMap sortedData;*

*for (friendMap::iterator i = database.begin(); i !=*

*database.end(); i++)*

*sortedData.insert(sortedMap::value\_type((\*i).second,*

*(\*i).first));*

*for\_each(sortedData.begin(), sortedData.end(), printSortedEntry);*

*cout << "end display by prefix" << endl;*

*}*

*int main() {*

*cout << "Telephone Directory sample program" << endl;*

*telephoneDirectory friends;*

*friends.addEntry("Samantha", 6342343);*

*friends.addEntry("Brenda", 5436546);*

*friends.addEntry("Fred", 7435423);*

*friends.addEntry("Allen", 6348723);*

*friends.displayDatabase();*

*friends.displayPrefix(634);*

*friends.displayByPrefix();*

*cout << "End of telephone directory sample program" << endl;*

*return 0;*

*}*

* 1. **Foydalanuvchining shablonlari**

Funksiya shablonlari bu *umumiy turlar* bilan ishlashi mumkin bo‘lgan maxsus funksiyalardir . Bu bizga har bir turdagi kodni to‘liq takrorlamasdan bir nechta turdagi yoki sinflarga moslashtirilishi mumkin bo‘lgan funksiyalar shablonini yaratishga imkon beradi.

C ++ da bunga *shablon parametrlari* yordamida erishish mumkin . Shablon parametrlari - bu argument sifatida turni o‘tkazish uchun ishlatiladigan parametrlarning maxsus turi: oddiy funksiyalar parametrlaridan qiymatlarni funksiyaga o‘tkazish uchun foydalanish mumkin bo‘lganidek, shablon parametrlari ham turlarga funksiyaga o‘tish imkonini beradi. Ushbu funksiya shablonlari ushbu parametrlardan boshqa odatdagidek foydalanishi mumkin.

Turi parametrlari bo‘lgan funksiyalar shablonini deklaratsiya qilish formati:

*template <class identifier> function\_declaration;*

*template <typename identifier> function\_declaration;*

Har ikki prototiplar orasidagi yagona farq kalit so‘z ham foydalanish hisoblanadi sinf yoki kalit so‘z TypeName . Uning ishlatilishi noaniq, chunki ikkala iboralar ham bir xil ma'noga ega va xuddi shunday harakat qilishadi.

Masalan, biz foydalanishimiz mumkin bo‘lgan ikkita obyektdan kattarog‘ini qaytaradigan shablon funksiyasini yaratish uchun:

| *template <class myType>*  *myType GetMax (myType a, myType b) {*  *return (a>b?a:b);*  *}* |  |
| --- | --- |

Bu erda biz shablon parametr sifatida myType yordamida shablon funksiyasini yaratdik. Ushbu shablon parametri hali aniqlanmagan turni ifodalaydi, ammo shablon funksiyasida odatdagi tur kabi ishlatilishi mumkin. Ko‘rinib turibdiki, GetMax funktsional shablonlari ushbu aniqlanmagan turning ikkita parametridan kattarog‘ini qaytaradi.

Ushbu funksiya shablonidan foydalanish uchun funksiyani chaqirish uchun quyidagi formatdan foydalanamiz:

function\_name <type> (parametrlar);

Masalan, Int tipidagi ikkita butun sonlarni taqqoslash uchun GetMax- ga qo‘ng‘iroq qilish :

| *int x,y;*  *GetMax <int> (x,y);* |  |
| --- | --- |

Kompilyator ushbu qo‘ng‘iroqni shablon funksiyasiga duch kelganda, u avtomatik ravishda shablon parametrlari sifatida o‘tgan turga ( bu holda int ) o‘tadigan turdagi MyType-ning har bir ko‘rinishini almashtiradigan funksiyani yaratishda foydalanadi. Ushbu jarayon kompilyator tomonidan avtomatik ravishda amalga oshiriladi va dasturchiga ko‘rinmaydi.

***Mana to‘liq misol:***

| *// function template*  *#include <iostream>*  *using namespace std;*  *template <class T>*  *T GetMax (T a, T b) {*  *T result;*  *result = (a>b)? a : b;*  *return (result);*  *}*  *int main () {*  *int i=5, j=6, k;*  *long l=10, m=5, n;*  *k=GetMax<int>(i,j);*  *n=GetMax<long>(l,m);*  *cout << k << endl;*  *cout << n << endl;*  *return 0;*  *}* | 6  10 |  |
| --- | --- | --- |

Bu holda biz T ni myType o‘rniga shablon parametrlari nomi sifatida ishlatdik, chunki u qisqaroq va aslida juda keng tarqalgan shablon parametrlari nomi. Lekin siz xohlagan har qanday identifikatordan foydalanishingiz mumkin.

Yuqoridagi misolda GetMax() funktsional shablonini ikki marta ishlatdik. Turi hujjatlar bilan birinchi marta int va turdagi hujjatlar bilan ikkinchi uzoq. Tuzuvchi dasturni o‘rnatdi va keyin har safar funksiyaning tegishli versiyasini chaqirdi.

Ko‘rinib turibdiki, T tipi GetMax () shablon funksiyasi doirasida ushbu turdagi yangi obyektlarni e'lon qilish uchun ham ishlatiladi:

|  | *T result;* |  |
| --- | --- | --- |

Shuning uchun, funksiya shablonini ma'lum bir tur bilan biriktirganda , natija a va b parametrlari bilan bir xil obyekt bo‘ladi . GetMax uchun parametr sifatida T umumiy tipi ishlatiladigan ushbu holatda , kompilyator avtomatik ravishda qaysi burchak turidagi qavs ichida uni aniq belgilash kerakligini aniqlay oladi (biz <int> va <long belgilashdan oldin qilganimiz kabi). > ). Shunday qilib, biz uning o‘rniga shunday yozsak bo‘ladi:

| *1 2* | *int i,j;*  *GetMax (i,j);* |  |
| --- | --- | --- |

Har ikki yildan i va j turi bo‘lgan int , va kompilyator avtomatik ravishda Shablon parametr faqat bo‘lishi mumkin, deb topish mumkin int . Ushbu noaniq usul aynan bir xil natijani beradi:

| *// function template II*  *#include <iostream>*  *using namespace std;*  *template <class T>*  *T GetMax (T a, T b) {*  *return (a>b?a:b);*  *}*  *int main () {*  *int i=5, j=6, k;*  *long l=10, m=5, n;*  *k=GetMax(i,j);*  *n=GetMax(l,m);*  *cout << k << endl;*  *cout << n << endl;*  *return 0;*  *}* | 6  10 |  |
| --- | --- | --- |

Bu holda, biz funksiya shablonni deb nomlangan xabarnoma qanday GetMax () ochiq burchakli qavs orasidagi turini ko‘rsatilgan holda <> . Tuzuvchi har bir qo‘ng‘iroqqa qanday tur kerakligini avtomatik ravishda aniqlaydi.

**1-mashq.** Ikkita butun son, kasr son, ikkita simvol va ikkita satrlarning maksimumini ekranga chop etuvchi dastur tuzing.

Biz quyidagiday to‘rtta qayta yuklanuvchi funksiyalaridan foydalanib mazkur dasturni tuzishimiz mumkin:

*int maxValue(int value1, int value2)*

*{*

*if (value1 > value2)*

*return value1;*

*else*

*return value2;*

*}*

*double maxValue(double value1, double value2)*

*{*

*if (value1 > value2)*

*return value1;*

*else*

*return value2;*

*}*

*char maxValue(char value1, char value2)*

*{*

*if (value1 > value2)*

*return value1;*

*else*

*return value2;*

*}*

*string maxValue(string value1, string value2)*

*{*

*if (value1 > value2)*

*return value1;*

*else*

*return value2;*

*}*

Mazkur to‘rtta funksiya bir-biriga o‘xshash, ularning farqi har bir funksiya boshqa bir turdagi qiymat qaytaradi. Birinchi funksiya int ni qaytaradi, ikkinchisining turi - double, uchinchisiniki - char, to‘rtinchi funksiyaning turi esa – string. Lekin xotiradan joyni iqtisod qilish va dastur ishini qulaylashtirish uchun umumiy holda biror bir universal tur yordamida dasturni tuzish imkoni mavjud, masalan quyidagicha:

*GenericType maxValue(GenericType value1, GenericType value2)*

*{*

*if (value1 > value2)*

*return value1;*

*else*

*return value2;*

*}*

Mazkur GenericType turi hamma turlarga to‘g‘ri keladi, ular int, double, char va string. C++ da funksiya shablonlarini universal tur bilan e’lon qilish imkoni mavjud. Ikki qiymat o‘rtasida maksimalini topuvchi dastur kodi ko‘rsatilgan.

*#include <iostream>*

*#include <string>*

*using namespace std;*

*template <typename T>*

*T* ***maxValue****(T value1, T value2)*

*{*

*if (value1 > value2)*

*return value1;*

*else*

*return value2;*

*}*

*int* ***main****()*

*{*

*cout << "Maksimum mejdu 1 i 3 raven " << maxValue(1, 3) << endl;*

*cout << "Maksimum mejdu 1.5 i 0.3 raven "*

*<< maxValue(1.5, 0.3) << endl;*

*cout << "Maksimum mejdu 'A' i 'N' raven "*

*<< maxValue('A', 'N') << endl;*

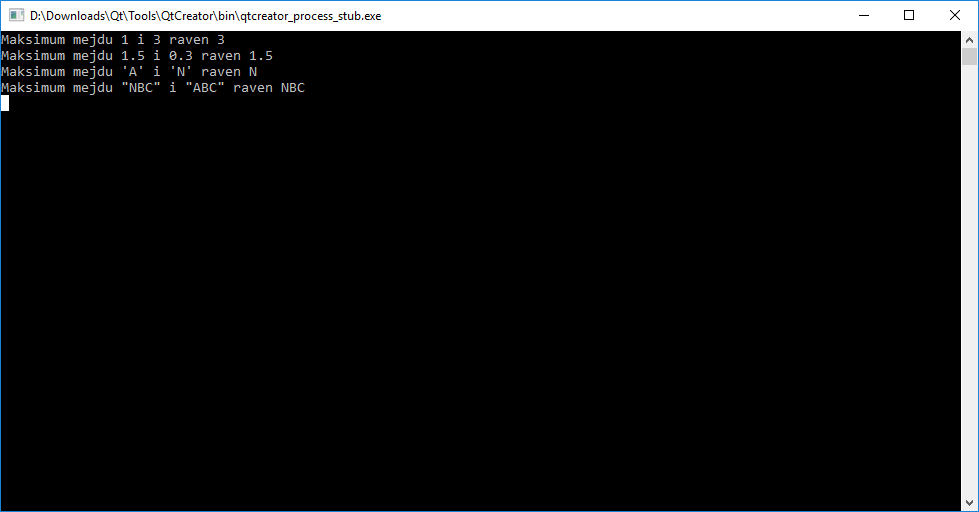
*cout << "Maksimum mejdu \"NBC\" i \"ABC\" raven "*

*<< maxValue(string("NBC"), string("ABC")) << endl;*

*return 0;*

*}*

**Natija:**



Shablon funksiyasini e’lon qilinishi template kalit so‘zi yordamida amalga oshiriladi. Har bir parametr oldidan typename kalit so‘zi yoki class kalit so‘zi qatnashishi va bu holda <typename typeParameter> yoki <class typeParameter> ishlatilishi maqsadga muvofiq. *Misol uchun*

template <typename T>

Funksiyaga argumentlar qiymat orqali yuborilishi ko‘rib chiqilgan. Quyidagi funksiya argumentlari havola orqali yuborilishi ko‘rib chiqilgan.

*#include <iostream>*

*#include <string>*

*using namespace std;*

*template<typename T>*

*T maxValue(const T& value1, const T& value2){*

*if (value1 > value2)*

*return value1;*

*else*

*return value2;*

*}*

*int main() {*

*cout << “Maksimum mejdu 1 i 3 raven “ << maxValue(1, 3) << endl;*

*cout << “Maksimum mejdu 1.5 i 0.3 raven “*

*<< maxValue(1.5, 0.3) << endl;*

*cout << “Maksimum mejdu ‘A’ i ‘N’ raven “*

*<< maxValue('A', 'N') << endl;*

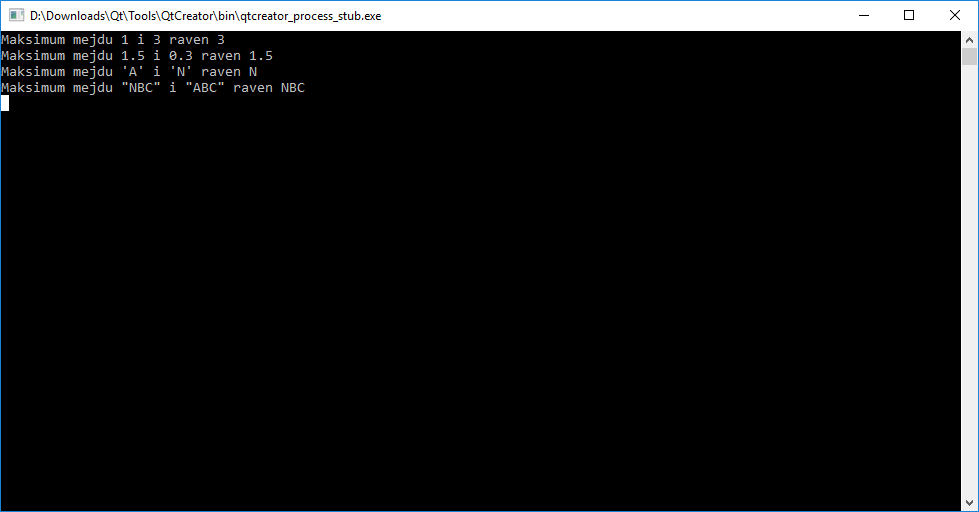
*cout << “Maksimum mejdu \”NBC\” i \”ABC\” raven “*

*<< maxValue(string(“NBC”), string(“ABC”)) << endl;*

*return 0;*

*}*

**Natija:**



**TOPSHIRIQLAR**

1. Mashina nomli class yarating va yaratilgan classga mashina markasi, ishlab chiqarilgan yil, yurgan masofasi kabi parametrlar yarating.Yaratilgan sinfdan info nomli metod yarating va chop eting. Yaratgan obyektlaringiz mashina1,mashina2, mashina3 kabi nomlang.

| **Input:** | **Output:** |
| --- | --- |
| Nexia  2008  30000 | Mashina markasi Nexia, ishlab chiqarilgan yil 2008, bosib o‘tgan masofa 30000 km |

1. Talaba nomli class yarating va yaratilgan classga talaba ismi, familyasi, kursi, manzili kabi parametrlar yarating. Yaratilgan sinfdan info nomli metod yarating va talaba haqidagi ma’lumotlarni chop eting. Yaratgan obyektlaringiz talaba1,talaba2, talaba3 kabi nomlang.

| **Input:** | **Output:** |
| --- | --- |
| Bahodir  Aliyev  2  Bulung‘ur  30000 | Talabaning ismi Bahodir, familyasi Aliyev 2-kurs talabasi, Yashash manzili Bulung‘ur |

1. Kompyuter nomli class yarating va yaratilgan classga modeli, operativ xotirasi kabi parametrlar yarating. Yaratilgan sinfdan info nomli metod yarating va kompyuter haqidagi ma’lumotlarni chop eting. Yaratgan obyektlaringiz komyuter1,kompyuter2 kabi nomlang.

| **Input:** | **Output:** |
| --- | --- |
| HP  16  Core i7  1TB | Kompyuter modeli HP, Operativ xotirasi 16 GB, Protsesor Core i7, vinti 1TB |

1. Ikki o‘zgaruvchili sinf yarating. Sinfga ikkita qiymat bering. Bu o‘zgaruvchilar qiymatlari yig‘indisini topadigan funksiyani va bu ikkita o‘zgaruvchining eng katta qiymatini topadigan funksiyani qo‘shing.

| **Input:** | **Output:** |
| --- | --- |
| 2000  3000 | Ikki sonning yig‘indisi: 5000  Ikki sonning kattasi: 3000 |

1. Ikki o‘zgaruvchili sinf yarating. Kirish parametrlari bo‘lgan konstruktor qo‘shing. Ikktita obyekt yarating va chop eting, keyin 2-obyektni o‘chiring.

| **Input:** | **Output:** |
| --- | --- |
| "Rustam",101  "Orzumurod",102 |  |