O’ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI

O’ZBEKISTON RESPUBLIKASI OLIY VA O’RTA MAHSUS TA’LIM VAZIRLIGI

MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

TOSHKENT MOLIYA INSTITUTI

DASTURLASH

fanidan

o’quv qo’llanma

Toshkent 2020

Mualliflar: B.Mo’minov, O.Mallayev, U.Yakubov. “Dasturlash” fanidan o’quv qo’llanma //“IQTISOD-MOLIYA” - 244 b.. Toshkent, 2020.

O’quv qo’llanma maqsadi – nazariy bilimlarni mustahkamlash hamda strukturali va ob’ektga yo’naltirilgan dasturlar yaratish va joriy etish amaliy ko’nikmalarini hosil qilishdan iborat.

O’quv qo’llanma dasturlashning asosiy tushunchalarini, ya’ni o’zgaruvchilar, funksiyalar, strukturalar va dinamik xotira bilan ishlash asoslarini yaratishga qaratilgan. Shu bilan birga sinflar, vorislik, amallarni qo’shimcha yuklash, istisnolardan foydalanib ob’ektli dasturlashga bag’ishlangan. Qo’llanma o’rta maxsus, kasb-hunar ta’limi va oliy o’quv yurtlari professor-o’qituvchilari va talabalari uchun mo’ljallangan.

O’quv qo’llanma Toshkent moliya instituti Kengashi tomonidan tasdiqlangan.

Mualliflar: B.Mo’minov - Muhammad Al-Xorazmiy nomidagi Toshkent Axborot Texnologiyalari Universiteti, “Informatika asoslari” kafedrasi mudiri v.b..

O.Mallayev - Muhammad Al-Xorazmiy nomidagi Toshkent Axborot Texnologiyalari Universiteti, “Informatika asoslari” kafedrasi kata o’qituvchisi.

U.Yakubov – Toshkent moliya instituti “Elektron tijorat va raqamli iqtisodiyot” kafedrasi mudiri v.b..

Taqrizchilar: B.Mo’minov - Muhammad Al-Xorazmiy nomidagi Toshkent Axborot Texnologiyalari Universiteti, “Axborot texnologiyalari” kafedrasi proffessori, t.f.d..

Yo.Ilhomova – Toshkent moliya instituti “Elektron tijorat va raqamli iqtisodiyot” kafedrasi dotsenti, i.f.n..

1. Kirish

Siz kompyuterni ish yoki vaqtichog'lik uchun ishlatishingiz mumkin. Ko'p insonlar kompyuterni xar kungi ishlari, elektron pul o'tkazish yoki kurs ishlarini yozish uchun ishlatadilar. Kompyuter shunday ishlar uchun keraklidir. Ular qo'lda bajariladigan zerikarli yumushlarni bajarishda, zerikishsiz va charchamasdan sonlar yig'indisini chiqarishda, sahifaga so'zlar yozishda foydalaniladi.

Kompyuterning moslashuvchanligi juda ajablanarli hodisadir. Shu mashina yana ro'yxat kitobini tekshirishda, kurs ishlari yaratishda, va o'yin o'ynashda ham foydalaniladi. Boshqa mashinalar bilan solishtirganda, ular kamroq vazifalar bajarishadi, mashina haydashadi va toasterda non qizartirishadi. Kompyuter ko'proq vazifalar bajaradigan, maxsus vazifalar buyuradigan dasturlarda ishlaydi.

Kompyuter aslida ma'lumotlar saqlashda(sonlar, so'zlar, raqamlar), qurilmalar bilan ulanishda(monitor, ovoz sistemasi va printer), va dastur bajarishda ishlatilinadi. Kompyuter dasturi kompyuterga nima qilish kearkligini batafsil detallarda, jarayonni yuzaga keltiradigan ketme-ket qadamlarni aytadi. Kompyuter moddiy qismlari va qo'shimcha qurilmalari qurilmaviy ta'minoti deb nomlanadi. Dasturlarni amalga oshiradigan qism dasturiy ta'minot deyiladi.

Bugungi kunda kompyuter dasturlari juda ham murakkabdir va u juda ham oddiy jarayonlardan tashkil topganiga ishonish qiyindir. Odatiy jarayonlar quyidagilar bo'lishi mumkin:

- ekrandagi shu joyda qizil nuqta qo'yadi;
- shu ikki raqamni qo'shadi;
- agar qiymat noijobiy bo'lsa, dastur boshqa aniq ko'rsatmalarda ishlaydi. Kompyuter illyuzion aniq bog'lanishlarni katta tezlikda bajaruvchi ulkan miqdordagi amallardan iboratdir.

Kompyuter dasturlarini dizayn qilish va uni ijrosini ta'minlash Dasturlash deyiladi. Bu kitobda, siz kompyuterni qanday dasturlashni o'rganasiz, shuningdek

amallar ijrosiga qanday yo’naltirishni ham.

Kompyuter o’yinlarini harakat va ovoz effektlari va fantaziyali va rasmli so’z jarayoni bilan yaratish kabi murakkab vazifa, jamoaviy yuqori malakali dasturchilarni talab qiladi. Sizning ilk dasturlashdagi muvaffaqiyatingiz maroqsiz bo’lishi mumkin. Bu kitobdagi siz o’rganadigan tushuncha va ko’nikmalar asosiy fundament bo’ladi va sizning ilk yaratagan dasturingiz siz bilgan dasturlar bilan raqobatlasha olmasa xafsalangiz pir bo’lmaydi.

Aslida, oddiy dasturlash jarayonlari ham ulkan zavq bag’ishlaydi. Kompyuter berilgan buyruqni aniq va tez amalga oshirishi ko’p soatli og’ir va zerikarli ishlardan so’ng, kichik o’zgartirishdan keyin tezlik bilan zaruriy to’g’irlashlash kiritishi va kompyuter sizning aqliy qobiliyatingiz ko’rinishi bo’lishi zavqli taasurotdir.

2.“Dasturlash” fanining mazmuni, predmeti va metodi

2.1 Kompyuter anatomiyasi

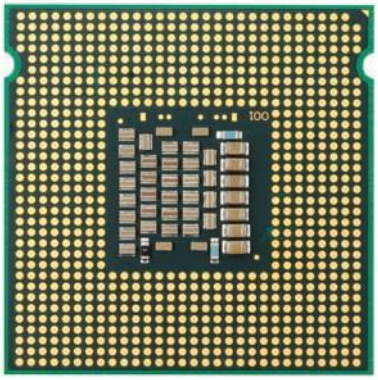
Dasturlash jarayonini tushunish uchun kompyuterni tashkil etgan qurilma bloklarini tushunishingiz kerak. Shaxsiy kompyuterni ko’rib chiqsak. Katta kompyuterlarni boshqa kompyuterlarnikidek dizayndagi ammo tezroq, kattaroq, yoki kuchliroq qislmari mavjud.

Kompyuterning yuragi Markaziy Protsessor(CPU) (1.01-rasmga qarang) Markaziy Protsessor yagona chipdan yoki kichik birlikdagi chiplardan iborat. Kompyuter chipi metal yoki plastik komponentli korpusdan, metal ulagichlardan iborat, uzatkichlar ichki qismi esa kremniydan iborat. Protsessor ichki qismi juda murakkab tuzilgan. MISOl uchun, Pentium chipi (qo’llanma yozilayotgan vaqtda shaxsiy kompyuterlar uchun mashxur protsessor sanalgan) bir qancha millionli tranzistor deb nomlanadigan tuzilma elementlaridan tashkil topgan.

Markaziy Protsessor dastur nazorat qilinadi va ma’lumot qayta ishlanadi. Bunda kompyuter dastur nazoratini amalga oshiradi va turgan o’rnini aniqlaydi; u

yana qo’shish, ayirish, ko’paytirish va bo’lish kabi arifmetik amallar ham bajaradi; tashqi xotira yoki qurilmadagi ma’lumotlarni orqaga qaytarib yozib saqlay oladi.

Kompyuter ma’lumot va dasturlarni saqlaydi. Ikki xil turdagi xotira mavjud. Xotira chiplaridan iborat, elektr quvvati bilan ta’minlangan o’zida ma’lumot saqlay oladigan elekr zanjirdan iborat xotira- Birlamchi xotira deyiladi. Qoshimcha xotira esa, kamroq qiymatga ega elektr sarflamaydigan Qattiq diskdir. Qatti disk aylanuvchi plastinkadan iborat, magnit material bilan qoplangan, plastinka oqimini aniqlab va o’zgartira oladigan o’quvchi/yozuvchi muhim qismlardan iborat (1.02 rasmga qarang).



1.01-rasm. Markaziy prosessor

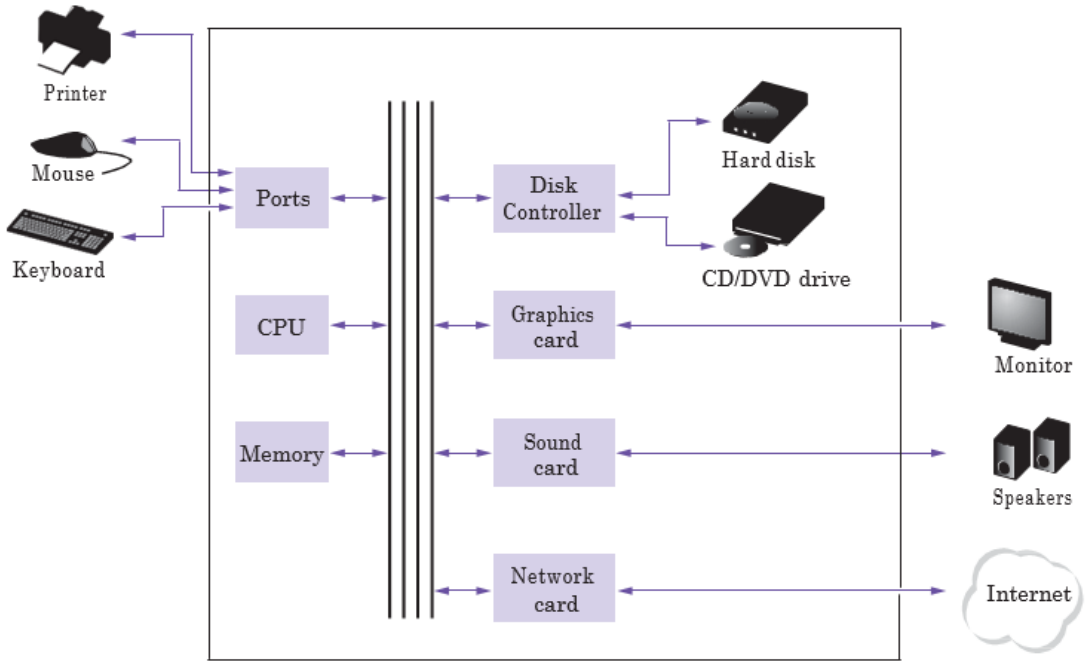
1.02-rasm. Qattiq disk

Dasturlar va ma’lumotlar odatda qattiq diskda saqlanadi va dastur ishga tushganda xotiradan olinadi. Dastur xotiraagi ma’lumotlarni yangilangandan

so’ng o’zgartirilgan ma’lumotlar qayta qattiq diskka yoziladi.

Insonlar bilan muloqotga kirihganda kompyuter qo’shimcha qurilmalarni talab qiladi. Kompyuter insonga ma’lumotlarni ekran, ovoz qurilmasi va printer orqali uzatadi. Inson ma’lumotni klaviatura yoki ko’rsatuvchi qurilma sichqoncha bilan kiritadi.

Ayrim kompyuterlar qismlari o’zida tashkil topgan, qolganlari bo’lsa tarmoq bilan bog’langan. Tarmoq kabellari sababli, komyuter markaziy xotiradan dastur va ma’lumotlarni o’qiydi va boshqa kompyuterlarga jo’nata oladi. Tarmoqqa ulangan komyuter foydalanuvchisi qaysi ma’lumot kompyuterda o’zi bo’lgani va qaysi biri tarmoqdan olingani aniq bo’ladi.



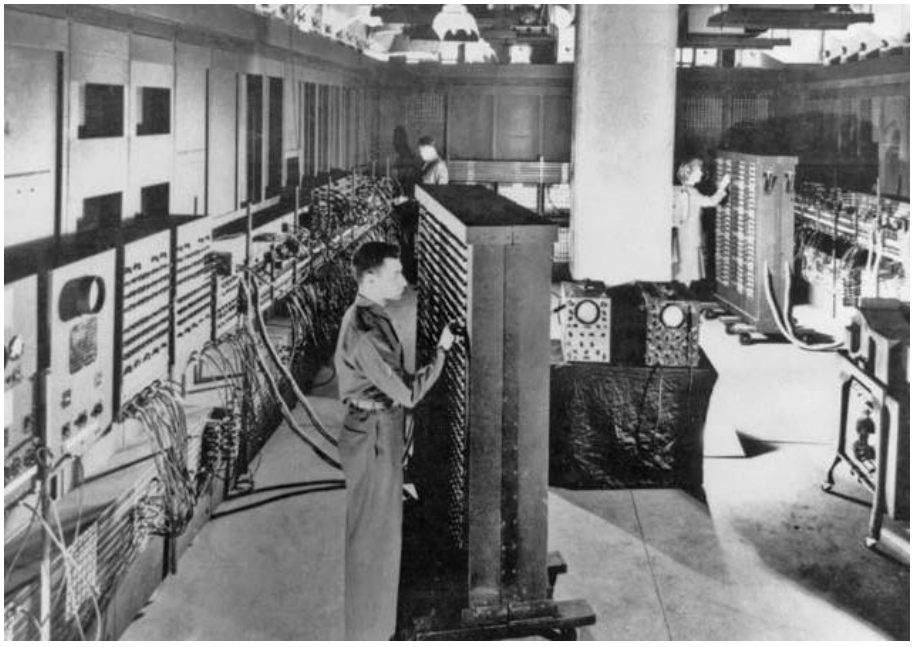
1.03-rasm. Shaxsiy kompyuterning sxematik tuzilishi

1.03-rasmda shaxsiy kompyuterning sxematik ko’rinish arxitekturasi berilgan. Dastur ko’rsatmalari va ma’lumotlari (matn, son, audio va videolar) qattiq diskda, optik disk DVD da yoki tarmoqning boshqa biror joyida saqlanadi. Dastur ishga tushganda u xotiradan olinadi va Markaziy protsessor uni o’qiydi. Markaziy Protsessor bir vaqtda bitta dastur ko’rsatmasini o’qiydi. Shu ko’rsatmaga binoan, Markaziy Protsessor ma’lumotni o’qiyd, qayta ishlaydi va

saqlaydi. Ayrim dastur ko’rsatmalari Markaziy Protsesorga, ekranda qayerga nuqta qo’yishni, bosib chiqarishni va ovoz qurilmasini ishga tushirishga buyruq beradi. Bu holatlar ko’p marotaba va yuqori tezlikda amalga oshgani sababli inson faqatgina ovoz va rasmlarni anglaydi. Ayrim dastur ko’rsatmalari klaviatura va sichqoncha orqali kiritilgan ma’lumot orqali o’qiladi. Dastur bu kiritilgan ko’rsatmalarni tahlil qiladi va kerakli ko’rsatmalarni amalga oshiradi.

Tasodifiy Fakt 1.1. ENIAK va hisoblashlar ibtidosi

ENIAK (yelektron nomerlovchi integrator va kompyuter) birinchi elektron kompyuterlar bo’lgan. U J.Presper va Jon Mauchliy tomonidan Pensilvaniya Universitetida 1946 yil tranzistorlar kashf qilinishidan 2 yil oldin kashf qilingan. Kompyuter katta xonaga joylashtirilgan va 18000 ga yaqin vakuumdan tashkil topgan kichkina shkaflardan tashkil topgan.(1.02 rasmga qarang) Vakuum trubalar kuniga bir nechtalab yonib ketgan. Maxsus shaxs trubalar to’ldirilgan idish bilan ishdan chiqqan trubalarni almashtirib turgan.



1.04-rasm. ENIAK

Kompyuter panellardagi birlashtiruvchi simlar bilan dasturlangan. Xar bir sim konfiguratsiasi tegishli kompyuter muammosi uchun o’rnatilgan. Kompyuter turli xil muammlor ustida ishlashi uchun simlar qayta ulangan. Qo’shma Shtatlar

Armiyasi ENIAKni ballistik jadvallar hisobi uchun, shamolga qarshi tezlik trayektoriyasi va atmosfera holatini aniqlab berishda foydalangan. Trayektoriyani hisoblash uchun ma’lum bir differensial tenglamaning raqamli yechimini topish shart edi. Shu sababli ham “raqamli integrator” deb nomlangan. ENIAK kabi mashinalar ixtiro qilinmasdan avval insonlar bu ishlarni qilishgan va 1950 yilgacha “kompyuter” so’zi ular uchun ishlatilgan. ENIAK keyinchali tinchlik yo’lida, Qo’shma Shtatlarda aholini ro’yxatga olishda ishlatilgan.

Tasodifiy Fakt 1.2 standardlovchi tashkilotlar

2 ta standardlovchi tashkilotlar Amerika Milliy Standardlash Instituti (ANSI) va Xalqaro Standardlovchi Tashkilot (ISO) hamkorlikda C++ tili uchun eng to’g’ri standard vujudga keltirishdi. Nima uchun standard kerak? Siz standardlashni foydasi bilan har kuni to’qnash kelasiz. Lampochka sotib olayotganda u uyingizdagi lampochka chanog’iga mos kelishini bilasiz.

Fakt shuki, siz qachondir nostandard fonar, lampochkalarini xarid qilsangiz standardlash qanchalik muhimligini bilib olasiz. Fonar Lampochkalarni qayta almashtirish qiyin va qimmat bo’lishi mumkin.

ANSI and ISO standardlash tashkilotlari mashina balonlari va kredit kartalari shaklidan to C++ dasturlash tiligacha xamma narsalarni standardlashni yo’lga qo’ygan ishlab chiqarish mutaxassislari birlashmasidir.

Bu siz bir sistemada o’rnatgan dasturni boshqa ishlab chiqaruvchi to’plamidagi boshqa dasturga qo’yganingizda u ishlashiga amin bo’lishingizdir.

2.2 Dasturlash muhiti bilan tanishish

Ko’p studentlar dasturchilarga kerak bo’ladigan qurilmalar ularga tanish bo’lgan dasturiy taminot qurilmalaridan faqr qilishini bilishadi. Siz alohida vaqt ajratib dasturiy muhit bilan tanishib chiqing. Chunki kompyuter tizimi keng farqlanadi, bu kitob sizga amal qilishingiz kerak bo’lgan bosqichlar yo’riqnomasini beradi.Amaliy labaratoriya ishlarida qatnashish yoki bilimga ega do’stingiz yo’rig’ini tinglash ham samaralidir.

1- bosqich. C++ yaralish muhitini boshlang.

Kompyuter tizimlari bu borada katta farq qiladi. Ko'p kompyuterlarda yozish va dasturlarni sinash mumkin bo'lgan bir integratsiya ishlab chiqish muhiti bor. Boshqa kompyuterlarda esa C++ yo'riqnomalarini kiritish uchun avval so'z muharririni ishga tushirish lozim, keyin esa konsol oynasi- ni ochish va bajarilishi kerak bo'lgan buyruqni kirgizishingiz kerak. Suiz muhit bilan ishlashni o'rganishingiz lozim.

2- bosqich. Oddiy dastur yozing.

Yangi dasturlash tilini yaratishdagi ilk dasturda ekranda oddiy salomlashuv so'zi "Salom, dunyo!".

Keling, bu an'ana amal qilaylik. Bu erda C++ "Salom, dunyo!" dasturi:

```
1.#include "stdafx.h"
2.#include <iostream>
3.using namespace std;
4.int main()
5.{
6.cout << "Salom, dunyo!" << endl;
7.getchar();
8.return 0;
9.}
```

Biz keyingi bo'limda ushbu dasturni ko'rib chiqamiz.

Qanday dasturiy muhitdan foydalanmang, dastur holatini muharrir oynasiga kiritishdan boshlaysiz.

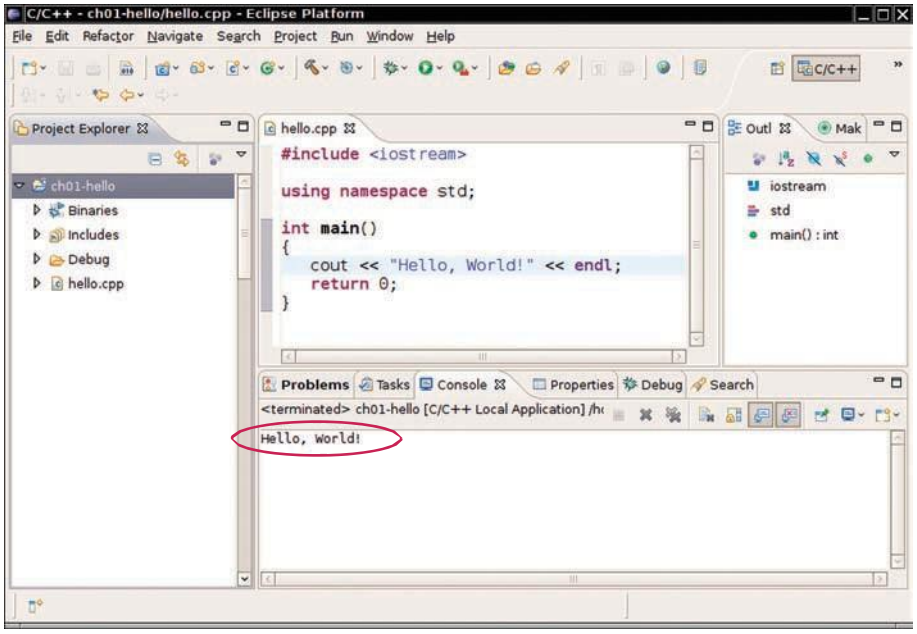
Yangi fayl oching va uni hyello.cpp, deb nomlang, yo'riqnomangiz uchun zarur bosqichlardan foydalaning. (Agar sizning muhitingiz loyiha nomini fayl nomiga qo'shib yozishni talab qilsa u holda siz hyello so'zini loyiha nomi deb olinng.) Dastur buyruqlarini tepada berilgandek aniq kirg'izing. Shu bilan bir qatoda, dasturdagi manbaa fayllarini elektron nusxasini toping va muharriringizga qo'ying.

Siz shu dasturni yozar ekansiz, turli belgilarga yaxshilab e'tibor bering, C++ nozik ish sanaladi. Siz xarflarni dastur satrida ko'ringanidek bosh va kichik xarflarda kirg'izishingiz shart. **main** yoki **endl** yoza olmaysiz. Agar siz e'tiborli

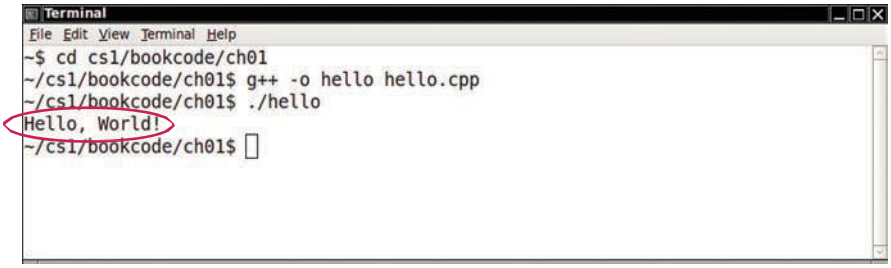
bo’lmasangiz xatoga yo’l qo’yasiz.

3- bosqich. Kompilyatsialash va dasturni ishga tushirish.

C++ dasturini yaratish va ishga tushirish jarayoni sizning dasturlash muhitingizga bog’liq. Ayrim kompleks rivojlantirish muhitida, siz oddiygina tugmachani bosasiz. Boshqa muhitda esa, siz buyruqlar kirg’izishingiz lozim. Siz dasturingizni sinovdan o’tkazishda Hello world! ekranni qaysi joyidadir paydo bo’ladi (1.05 va 1.06 rasmga qarang).



1.05-rasm. hello dasturini ishga tushirish jarayoni



1.06-rasm. hello dasturini window oynasida kompilyasiyalash va ishga tushirish

Dasturingiz tuzilayotganda uning negizida nima borligi muhimdir. Birinchidan, First, kompilyator C++ manbaa kodini (siz kiritgan so’zlar) mashina

ko’rsatmalariga o’girib beradi. Mashina kodi siz yozgan so’zni kodga o’girib berilgan shaklidir. Dasturni faqatgina ishga tushirish yetarli bo’lmaydi. Kompyuter oynasida bir tizimni ko’rinishi uchunham, kam darajada bo’lsa ham faollik muhim. C++ rivoj- lantirihs muhiti amaliyotchilari cout va uning vazifalarini o’z ichiga olgan kutub- xona bilan ta’minlab beradi. Kutubxona bu bowqa bir inson tomonidan dastur- langan va tarjima qilingan, siz qo’llashingiz uchun tayyor dastur kodlar va jamlanmasidir. (Qo’shimcha murakkab dasturlar 1 dan ortiq mashina kodi fayli va ku- tubxonadan iborat) Birlashtirgich nomli dastur mashina kodini va C++ kutubxonasidan kerakli qismlarni birlashtirib amalga oshgan faylni yuzaga keltiradi. (1.07-rasmda bu boshqichlarni ko’rinishi tsvirlangan) Amalga oshgan fayl kompyuter tizimiga bog’liq ravishda hyello.yexe yoki hyello deb nomlanadi. Siz amalga oshgan faylni C++ rivojlantirish dasturidan chiqib ketganingizda ham ishga tushirishingiz mumkin.

4- bosqich. Ishingizni tashkillashtiring.

Dasturchi sifatida, siz dastur tuzasiz, sinaysiz va takomillashtirasiz Siz dasturlaringizni fayllarda saqlaysiz. Fayllarni nomlari va qonuniy nomlari qoidalar tizimi biri ikkinchisidan farq qiladi. Ayrim tizimlar fayl nomlari orasida bo’sh joy qoldirishga ruxsat beradi ayrimlari esa yo’q. Ayrimlarida kat- ta va kichik harflar farqlanadi, ayrimlarida esa yo’q. Ko’p C++ kompilyatorlar C++ fayllari **.cpp**, **.cxx**, **.cc**, yoki **.c** bilan tugashi talab qilinadi, masalan **test.cpp**.

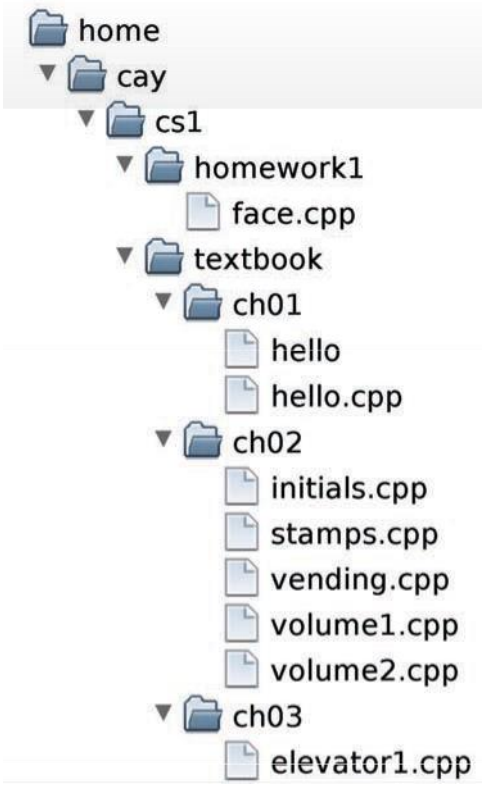
Papkalar o’z ichiga fayllarni oladi va yana ichida fayllar va papkalar bor boshqa fayllarni ham saqlay oladi. (1.08-rasm) Bu iyerarxia katta bo’lishi mumkin, va siz uning barcha bo’limlari bilan tanish bo’lmasligingiz mumkin.

1.07-rasm. Amalga oshgan fayl manba kodi

Biroq siz ishingizni tashkil qilishingiz uchun papkalar yaratishingiz mumkin. Har bir dasturlash darsi uchun alohida papka ochish to’g’ri fikrdir. Papka ichida esa, xar bir topshiriq uchun alohida papka oching.

Ayrim dasturlash muhiti papkani o’zingiz joylashtirmasangiz standart manzilga saqlaydi. U holda siz papka joylashgan joyini aniqlashingiz kerak bo’ladi.

Iyerarxia papkasida faylingiz qayerga joylashganini bilishingizga ishonch hosil qiling. Bu ma’lumot papkalarni guruhlayotganingizda va zahira nusxa olayotganingizda muhimdir.



1.08-rasm. Ierarxiya fayli

3. Dasturlash tillarining tuzilmasi

3.1 Ilk dasturingiz tahlili

Bu bo’limda ilk C++ dasturini batafsil tahlil qilamiz. Quyida yana manbaa kodi:

```
10.    #include "stdafx.h"
11.    #include <iostream>
12.    using namespace std;
13.    int main()
14.    {
15.    cout << "Salom, dunyo!" << endl;
16.    getchar();
17.    return 0;
18.    }
```

using namespace std; kompilyatorga "standart nomlanish joyi" ishlatish kerakligini aytadi. Nomlanish joyi katta dasturlarda nomlanishdagi nizolarni bartaraf etadi. Siz nomlanish joyidan xavotir olmasangiz ham bo'ladi. Bu kitobda ishlab chiqadigan dasturingiz uchun standart nom qo'yasiz. xar bir yzadigan dasturingiz boshiga using namespace std; ni yozing va oxirida esa #include ko'rsatmasini yozing.

Tuzilishi

```
int main()

{

...

return 0;

}
```

main deb nomlanuvchi funksiasi butun sonni 0 qiymatda qaytaradi (bunda C++ int da deb nomlanuvchi bo'laklanmagan butun son) Bu qiymat dastur muvaffaqiyatli yakunlanganini ifodalaydi. Funksia bu belgilangan vazifani bajaruvchi dasturlash yo'riqnomasidir. Har bir C++ dasturining asosiy funksiasi bor.

Xozir esa, oddiy dastur yozish uchun kerak bo'ladigan qismlarni tahlil qilsak to'g'ri bo'ladi. Kodni amalga oshirmoqchi bo'lgan main funksiasiga kiriting.

Qiymatni ekranda ko'rsatish uchun cout nomli asos va << operator(ayrim paytlarda o'rnatilgan operator)) dan foydalanishingiz mumkin. Masalan,

```
cout << 39 + 3;
```

42 soni ekranda ko'rinadi.

3.2 Xatolar

Dasturlash tili juda ham muhim konvensialarga asoslanadi. Siz bir inson bilan muloqot qilayotganingizda 1 yoki 2 so'zni o'tkazib yuborsangiz yoki tushirib qoldirsangiz siz bilan muloqotdagi inson nima demoqchi ekanligingizni tushuna oladi. Ammo C++ da xatoga yo'l qo'ysangiz kompilyator noto'g'ri tushunchani qabul qilmaydi. (Bu xasqiqatda yaxshi narss, agar kompilyator noto'g'ri tushunchani qabul qilganda, u natijani ham noto'gri taqdim qilar edi. Bu esa falokatli oqibatlarga olib kelar edi.) Bu bo'limda siz dasturingizdagi xatolarni qanday bartaraf etishni o'rganasiz.

hello.cpp dasturi bilan tajriba qilamiz. Biz quyidagi xatolarga yo'l qo'ysak nima sodir bo'lar edi.

```
cout << "Hello, World!" << endl; cout
<< "Hello, World! << endl; cout << "Hollo,World!"<<endl;
```

Birinchi holatda, kompilyator orqali nimani nazarda tutayotganingizni tushunmaganligidan arz qiladi. Yo'l qo'yilgan xatolikning aniq ta'rifi kompilyatorga bog'liq bo'ladi. Bu “Undefined symbol cot” kabi ko'rinishda ham bo'lishi mumkin. Bu COMPILE TIME xatolik yoki sintaksis xatolik sanaladi. Imlo qoidasiga yo'l qo'yilsa ham kompilyator uni topadi. Agar kompilyator 1 yoki undan ko'p xatolikni topsa, u holda dasturni texnika tiliga o'girmaydi va natijada ishga tushiriladigan dastur ham yaratilmaydi. Siz xatolikni bartaraf eti uni boshqatdan to'plashingiz kerak bo'ladi. Odatda ilk muvaffaqiyatli kompilyatisaga erishishdan oldin COPMILE TIME xatoliknibartaraf etishdagi bir qancha jarayonlardan o'tiladi.

Agar kompilyator xatolikni aniqlasa u osonlikcha to'xtamaydi va rad etmaydi. U aniqlagan xatolikni hammasini ko'rsatadi, siz bunda hamma xatolikni bittada to'g'irlab olishingiz mumkin. Ayrim paytlarda bitta xatoni o'zi ham dasturni

ishdan chiqarishi mumkin. Bunday xatolik ikkinchi satrda ham uchrashi mumkin. Dasturchi yopuvchi qavs belgisini ishdan chiqarsa kompilyator satr oxirini qidirishda davom etadi. Bunday holatlarda kompilyator qo'shni qatorlarda soxta xatolikni ko'rsatadi. Siz kerakli qatorlardagi xatolikni to'g'irlab so'ng qaytadan kompilyatsialashingiz kerak.

Bu RUN TIME ERROR hisoblanadi. Dastur gap tuzilishi jihatdan xatolikni topadi va nimadir bajaradi, ammo taxmin qilingan ishni bajarmaydi. Kompilyator xatolikni topa olmaydi, ammo dastur ishga tushganda uni tozalashga majbur, uni tekshiradi va chiqarilishiga yaxshilab e'tibor beradi.

RUN-TIME ERROR sabali dastur mantiqiy nuqson aniqlaydi va bu

nuqsonlar mantiqiy xatolik deyiladi. Ayrim RUN TIME ERROR xatoliklari jiddiy hisoblanadiki ayrim istisnolar keltirib chiqarishga ham sababchi bo'ladi. Protsessordagi xabar xato xabar sababli dasturni tugatilishiga olib keladi.Misol uchun dasturingiz cout << 1 / 0; bo'lsa "nol bilan ajratish" istisnosi bilan tugatilinadi.

3.3 Muammo yechimi: Algoritm konstruksiyasi

Siz tez kunda hisoblarni va qaror qabul qilishni C++da qanday dasturlashni o’rganasiz.Biroq keyingi bobdagi hisoblarni amalda qo’llash mexanizmini ko’rib chiqishdan oldin keling, ijrodan keyin keladigan rejalashtirish jarayonini ko’rib chiqamiz.Balki sizga mos umr yo’ldosh topib beruvchi kmpyuterlashagn xizmat uchun sizni undaydigan e’longa duch kelgandirsiz.

U qanday ishlashi mumkinligi to’g’risida o’ylab ko’ring.Siz anketa to’ldirasiz va uni jo’natasiz. Boshqalar ham shunday qiladilar. Ma’lumotlar kompyuter tomonidan ishlab chiqiladi. Kompyuter sizga eng mos shaxsni topish vazifasini uddalay oladi deb o’ylash to’g’rimi? Deylik kompyuter emas ukangizda barcha anketlar bor.Unga qanday ko’rsatmalar bera olardingiz.siz unga "Konkida uchishni va internetda o’tirishni yoqtiradigan juda chiroyli bo’lgan qarama-qarshi jinsdagi shaxsni top" , deb aytolmaysiz. Go’zal chehra borasida obyektiv standart

yo’q va ukangizning fikri (yoki raqamli rasmni tahlil qilgan kompyuterning dasturining fikri)siznikidan farqli bo’lishi mumkin. Agar siz biror kimsaga muammoniu hal etish chun yozma ko’rsatma bera olmasangiz, kompyuter ham buni hyech qanday sehr yordamida yecha olmasligini iloji yo’q.Kompyuter faqatgina unga nima qilishni aysangiz shuni bajara oladi. U zerikmasdan yoki charchamasdan vazifani tezroq bajaradi. Endi quyidagi sarmoya borasidagi muammoni ko’rib chiqamiz:Siz bank hisob raqamingizga yiliga 5% foyda qiladigan $10,000 miqdordagi pulni qo’ydingiz. Dastlabki miqdor ikki baravar ko’payishi uchun hisob balansi uchun qancha yil talab etiladi? Ushbu muammoni qo’llar yordamida yecha olasizmi? Albatta, siz balansni quyidagicha hisoblaysiz: yil foyda balans

year	interest	balance
0		10000
1	10000.00 x 0.05 = 500.00	10000.00 + 500.00 = 10500.00
2	10500.00 x 0.05 = 525.00	10500.00 + 525.00 = 11025.00
3	11025.00 x 0.05 = 551.25	11025.00 + 551.25 = 11576.25
4	11576.25 x 0.05 = 578.81	11576.25 + 578.81 = 12155.06

Balans kamida $20,000 miqdoriga yetmaguncha siz hisobni davom ettirasiz.Yil ustunidagi oxirgi raqam javob bo’ladi.

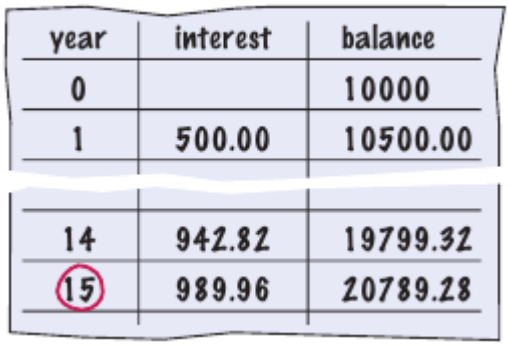
Albatta, bu xisoblashni amalga oshirish siz va ukangiz uchun juda zerikarlidir. Lekin kompyuterlar takrorlanuvchi hisoblarni tez va mukammal bajarishda ustasi farangdrlar.Kompyuter uchun muhimi bu yechimni topishda bosqichlarning tavsifidir.Har qaysi qadam tahmindan uzoqda aniq va lo’nda bo’lishi shart. Mana quyidagicha tavsif:

Yil qiymatini 0 bilan, foyda va $10,000 lik balans uchun ustunlardan boshlang

year	interest	balance
0		10000

Balans kamida $20,000ga yetmaguncha quyidagi qadamlarni qaytaring. Yil

qiymatiga 1 ni qo’shing. Foydani balance x 0.05 (ya’ni 5% foyda) deb qisoblang.Foydani balansga qo’shing.



year	interest	balance
0		10000
1	500.00	10500.00
14	942.82	19799.32
15	989.96	20789.28

Oxirgi yil qiymati javobligi to’g’risida xisobot bering.

Albatta, bu qadamlar kompyuter tushuna oladigan tilda emas, lekin siz tez orada ularni C++da qanday ifoda etishni o’rganib olasiz.Bunday norasmiy tavsif psevdokod deb ataladi. Psevdokod uchun qat’iy talablar yo’q, chunki uni kompyuter emas odmlar o’qishadi.

Bu yerda bu kitobda foydlanadigan psevdokod operatorlarining turlari berilgan:

Qiymat qanday berilgan va o’rnatilganini tasvirlaydigan quyidagi kabi operatorlardan foydalaning:

```
total cost = purchase price + operating cost
```

yoki

```
Multiply the balance value by 1.05.
```

yoki

```
Remove the first and last character from the word.
```

Siz qarorlarni va takrorlashlarni quyidagicha tasvirlashingiz mumkin:

```
If total cost 1 < total cost 2
```

```
While the balance is less than $20,000 For each picture in the sequence
```

Qaysi operator tanlanishi yoki takrorlanishi kerakligini belgilash uchun satr boshidan foydalaning:

```
For each car
operating cost = 10 x annual fuel cost total cost =
purchase price + operating cost
```

Bu yerda satr boshi ikkala operator har qaysi mashina uchun bajarilishi kerakligini ko’rsatyapti.

•Natijani ushbu operatorlar kabi ko’rsating:

```
Choose car1.
Report the final year value as the answer.
```

Aniq so’zma- so’z aytish muhim emas. Muhimi bu psevdokod bosqichlarni tasvirlashidadir, ya’ni:

- lo’nda
- amalga oshadigan
- yakunlanadigan

Har qaysi bosqichda nima qilish va keyin qayerga borish kerakligi to’g’risida aniq ko’rsatmalar bo’lsa, bu aniq metoddir.Taxmin va yaratuvchanlikka joy yo’q.Har qaysi bosqich amalda bajarilganda bu usul amalga oshuvchi qiymatdan emas yaqin yillarda olinadigan haqiqiy foiz qiymatidan foydalanish so’ralsa, bizning usul amalga oshadigan metod bo’lmaydi, chunki aynan o’sha foyda qiymatini hyech kim hyech qanday usulda bila olmaydi. Agar usul nihoyasiga yetsa u tugallanadigan metoddir.Bizning misolda metod uzluksiz davom etmasligini ko’rish ozgina fikrlashni talab etadi: Har qaysi bosqichda balans kamida $500 miqdoriga oshmoqda va shunday qilib u natijada $20,000 miqdoriga yetishi shart.

Aniq, amalga oshuvchi va yakunlanuvchi bosqichlarning ketma - ketligi algoritm deyiladi. Sarmoyamizdagi muammoni yechish uchun algoritm topdik, va shunday qilib kompyuterni dasturlash orqali yechim topishimiz mumkin.Algoritmni mavjudligi vazifani dasturlashning muhim talabidir.Dasturlashni boshlashdan oldin birinchi bo’lib siz xal qilishni xoxlagan

vazifa uchun algoritm yaratishingiz va tavsiflashingiz kerak.

Nazorat savollari:

1. Ayni paytda ishlamayotgan dastur qayerda saqlanadi?
2. Kompyuterning qaysi qismi arifmetik amallar qo’shish va ko’paytirishni amalga oshiradi?
3. Kompilyator qurilmaviy ta’minot qismimi yoki dasturiy ta’minot qismimi?
4. C++ ning eng muhim vazifasi nima?

4. Tarmoqlanish operatorlari. Takrorlanish operatorlari

4.1 O’zgaruvchilar

Sizning dasturingiz xisoblashni amalga oshirganda siz qiymatlarni shunday saqlashni xoxlaysizki, keyinroq ulardan foydalana olish uchun. C++ dasturida siz qiymatlarni saqlash uchun o’zgaruvchilvrdan foydalanasiz. Ushbu bo’limda siz o’zgaruvchilarni qanday qilib aniqlash va ulardan qanday foydalanishni o’rganasiz.

4.1.1 O’zgaruvchiga ta’rif

```
int cans_per_pack = 6;
```

O’zgaruvchi kompyut xotirasida saqlab qoluvchi qurilma yacheykasidir. Xar bir o’zgaruvchi o’z nomiga ega va qiymatdan iborat bo’ladi.

O’zgaruvchini aniqlayotganda siz odatda uni belgilashni xoxlaysiz. Ya’ni siz o’zgaruvchida saqlanishi kerak bo’lgan qiymatni belgilaysiz. Yana bir bor shu o’zgaruvchining ta’rifini ko’rib chiqaylik:

```
int cans_per_pack = 6;
```

Ma’lum bir turdagi transport vositasining (yengil avto, mototakrorlash yoki elektr avtomobil kabi) muayyan turi uchun mashinalar to’xtash joyida joy cheklanganidek, C++ dasturida o’zgaruvchi muayyan turdagi ma’lumotni saqlaydi. C++ dasturi deyarli ma’lumotlar turlari: raqamlar, matnli satrlar, fayllar, ma’lumotlar va boshqalarni qo’llaydi.Har gal o’zgaruvchini aniqlaganda siz turini qo’rstishingiz shart.

4.1.2 Sonlarning turi

C++ dasturida bir qancha har hil sonlar turi bor.Siz C++ dasturida int deb ataluvchi butun son turidan kasr qismisiz butun sonni anglatish uchun foydalanasiz.Masalan, bankaning istalagan to’plamidagi bankalar soni butun sonda bo’lishi kerak - bankaning soni kasr sonda bo’lmaydi.

1-jadval

Raqam	Tur	Izoh algebrik son literal
6	int	Butun son kasr qismga ega emas
-6	int	Butun son manfiy bo'lishi mumkin
0	int	Nol soni butundir.
0.5	double	kasrli son double turiga ega.
1.0	double	Butun son kasr bilan. .0 soni double turiga ega.
1E6	double	Son ko'rsatkichli sanoq sistemasida: 1×10^6 or 1000000.
ko'rsatkichli sanoq sistemasidagi son har doim double turiga ega. 2.96E-		
2	double	manfiy ko'rsatkich: $2.96 \times 10^{-2} = 2.96 / 100 = 0.0296$
100,000		Xato: O'nlikni ajratish uchun verguldan foydalanmang.

4.1.3 O'zgaruvchilarning nomi

O'zgaruvchini aniqlaganda siz uning maqsadini aks ettirgan nomni tanlashingiz kerak.Misol uchun, cv qisqa nomdan ko'ra uni t asvirlovchi nom can_volume dan foydalanish afzalroqdir.

C++ dasturida o'zgaruvchining nomini yozishning bir qancha oddiy qoidalari mavjud:

- O'zgaruvchining nomi harf yoki pastki chiziq belgisidan boshlanishi shart.Qolgan belgilar harflar, sonlar yoki pastki chiziq belgilari bo'lishi shart.
- $ yoki % belg ilarig a o'xshash belg ilarni ishlat a olmaysiz.Nomlarni yozishda joy qoldirish ta'qiqlangan. can_volumedagi kabi pastki chiziqdan foydala olasiz.
- O'zgaruvchining nomi klaviatura registrini inobatga oladi ya'ni Can_volume va can_volume xar- hildir.Shuning uchun o'zgaruvchining nomi uchun kichik harflardan foydalanish ma'qul fikr.
- double yoki return kabi zahira so'zlardan nomlar sifatida foydlana

olmaysiz;Bu so’zlar C++ dasturida faqatgina maxsus ma’nolari uchun zahiraga olingan.

4.1.4 O’zlashtirish operatori

O’zgaruvchiga yangi qiymatni joylashtirish siz uchun o’zlashtirish operatoridan foydalanasiz. Masalan:

```
cans_per_pack = 8;
```

O’zlashtirish operatorining chap tomoni o’zgaruvchidan tashkil topadi. O’ng tomoni esa qiymatga ega ifoda hisoblanadi. Ana o’sha qiymat o’zgaruvchida saqlanib uning avvalgi tarkibini qayta yozadi.

O’zgaruvchining ta’rifi va o’zlashtirish operatori o’rtasida muhim farq bor:

```
int cans_per_pack = 6; // O’zgaruvchining ta’rifi
...
cans_per_pack = 8; //O’zlashtirish operatori
```

Brinchi cans_per_pack ifodasi bu ta’rifdir. Bu int turidagi yangi o’zgaruvchini yaratish, unga cans_per_pack nomini byerish va uni 6 bilan belgilash qo’rsatmasidir.Ikkinchi ifoda o’zlashtirish operatori bo’lib, mavjud bo’lgan o’zgaruvchi cans_per_packning tarkiblarini boshqa qiymat bilash almashtirish ko’rsatmasidir.

= belgisi chap tomon o’ng tomonga tengligini anglatmaydi. O’ng tomondagi ifoda hisoblanadi va uning qiymati chap tomondagi o’zgaruvchiga joylashtiriladi. Bu o’zlashtirish operatorini algebrada tenglikni ifoda etish uchun ishlatiladigan = belgi bilan chalkashtirmang.O’zlashtirish operatori biror narsani bajarish, ya’ni qiymatni o’zgaruvchiga joylashtirish qo’rsatmatisidir. Matematik tenglikikki qiymat teng ekanligini ifoda etadi.

Masalan C++ dasturida , bunday yozish qonuniydir:

```
total_volume = total_volume + 2;
```

4.1.5 Doimiylar

O’zgaruvchi const zahira so’zi bilan belgilanganda, uning qiymati hyech qachon o’zgarmaydi. Doimiylar muntazam o’zgaruvchilardan hosil bo’ladi:

```
const double BOTTLE_VOLUME = 2;
```

Dasturinigizda raqamli qiymatlarining ma’nosini tushuntirish uchun nomiga ega doimiylardan foydalanish dasturlashning yaxshi uslubidir.Masalan, quyidagi ifodlarni taqqoslang

```
double total_volume = bottles * 2;
```

va

```
double total_volume = bottles * BOTTLE_VOLUME;
```

Birinchi ifodani o’quvchi dasturchi 2 sonining ahamiyatini tushunmasligi mumkin.Ikkinchisi esa ancha tushunarliroq.

4.1.6 Izohlar

Dasturingiz murakkablashgan sari, siz kodingizni o’quvchi shaxslar uchun unga izoh va tushuntirishlar qo’shishingiz kerak.Mana, bir misol:

```
const double CAN_VOLUME = 0.355; // Liters in a 12-ounce can
```

Bu izoh o’quvchi shaxsga 0.355 qiymatining ahamiyatliligini tushuntiradi.Kompilyator izohlarni umuman ishlab chiqmaydi. U // cheklovchidan qator oxirigacha hyech narsani inobatga olmaydi.

4.2 Arifmetika

Quyidagi bo’limda siz C++dasturida arifmetik va matematik hisob - kitoblarni qanday amalga oshirishni o’rganasiz.

4.2.1 Arifmetik amallar

C++ dasturi kalkulyator kabi bir xil asosiy to’rtta arifmetik amallarni: qo’shish, ayrish, ko’paytirish va bo’lishlarini amalga oshiradi, lekin ko’paytirish va

qo’shish amallari uchun boshqacha belgidan foydalanadi. Ko’paytirishni bildirish uchun siz a * b yozishingiz shart .Matematikadan farqli o’laroq, siz a b, a . b yoki a × b ko’rinishda yozolmaysiz . Shu singari bo’lish har doim a / kabi belg ilanadi.Uni hyech qachon a ÷ yoki kasr chizig’i bilan belgilamang.

4.2.2 Ko’payuvchi va kamayuvchi

O’zgaruvchini 1ni qo’shib yoki ayrib o’zgartirish shunday keng tarqalganki,uning uchun maxsus qisqartma mavjud, ya’ni

```
counter++;
counter--;
```

++ oshirish operatori C++ dasturiy tiliga o’z nomini berdi. C++ C tilining asta - sekin takomillashishidir.

4.2.3 Qoldiqsiz bo’linish va qoldiq

Bo’lish siz kutganiningizdek ishlaydi, xar ehtimolga qarshi qatnashgan sonlarning kamida bittasi suzuvchi nuqtali son bo’lgani kabi.Ya’ni, 7.0 / 4.0, 7 /

4.0, va 7.0 / 4 barchasi 1.75 beradi. Biroq, agar ikkala son butun bo’lsa, bo’linma natijasi har doim qoldiq tushirib qoldirilib butun son bo’ladi.Ya’ni,

```
7 / 4
```

bo’linmaning natijasi 1ga teng, chunki 7ni 4ga bo’lsak, 1butun 3 qoldiq (u tushirib qoldiriladi) natijani beradi. Bu noaniq dasturlash xatosining negizi bo’lishi mumkin.

Agar sizni faqat qoldiq qiziqtirsa, % operatoridan foydalaning:

```
7 % 4
```

butun sonlar bo’linmasining ya’ni 7ni 4ga bo’lganda natijaning qoldig’i 3ga teng. %belgisining algebrada analogi yo’q.U / belgisiga o’xshash bo’lgani uchun va qoldiqli amal bo’lish bilan aloqadarligi sababli tanlangan. Ushbu operator modulli bo’lish operatori deb nomlanadi.(Ba’zi odamlar uni modulo yoki mod deb atashadi.)Ba’zi kalkulyatorlarda uchratganingizdek uning foizli amali bilan hyech

qanday aloqasi yo’q. Quyida / va % amallarining qo’llanilishiga oddiy namuna berilgan .

Aytaylik, siz jamg’arma qutingizda bir qancha pen bor

```
int pennies = 1729;
```

Siz qiymatni dollarda va sentda hisoblashni hohlaysiz Siz dollarga 100 ga bo’lish butun bo’linma orqali ega bo’lasiz.

```
int dollars = pennies / 100; // Sets dollars to 17
```

4.2.4 Suzuvchi nuqtali sonlarni butun sonlarga aylantirish

Suzuvchi nuqtali qiymat butun o’zgaruvchiga o’zlashtirilganda, kasr qismi tushirilib qoldiriladi:

```
double price = 2.55;
int dollars = price; // Sets dollars to 2
```

Kasr qismni tushirib qoldirish bu har doim ham siz istagan narsa emas.Ko’pincha siz uni eng yaqin butun songa yaxlitlatlashni istaysiz.Musbat suzuvchi nuqtali qiymatni eng yaqin butun songa yaxlitlash uchun 0.5ni qo’shing va sungra uni butun songa aylantiring:

```
int dollars = price + 0.5; // eng yaqin songa yaxlitlanishlar
```

Bizning misolda, 0.5ni qo’shish 2.5dan katta barcha qiymatlarni 3dan katta qiymatlarga aylantiradi.Xususan, 2.55 qiymat 3.05ga aylantiriladi keyinchalik u 3 qiymatiga yaxlitlanadi.(manfiy suzuvchi nuqtali qiymat uchun siz 0.5ni ayirasiz.)Chunki yaxlitlash xatolarning asosiy sababidir,sizning kompilyatoringiz suzuvchi nuqtali qiymatni butun son o’zgaruvchiga o’zlashtirish xavfli ekanligini ogohlantirishi mumkin.

4.2.5 Darajalar va ildizlar

2-jadval

Жадвал 5 Арифметик ифодалар

Математик ифодалар	C++да ифодалар	Изоҳлар
$\frac{x + y}{2}$	(x + y) / 2	қавслар зарур; x + y / 2 computes $x + \frac{y}{2}$.
$\frac{xy}{2}$	x * y / 2	Қавслар зарур эмас; бир хил устунликдаги операторлар чапдан ўнга қараб ҳисоблайдиоperators .
$\left(1 + \frac{r}{100}\right)^n$	pow(1 + r / 100, n)	Дастурингиз юқорисига #include <cmath> ни қўшишни ёдда тутинг .
$\sqrt{a^2 + b^2}$	sqrt(a * a + b * b)	a * a ифодаси pow(a, 2)ифодасидан кўра соддароқ.
$\frac{i + j + k}{3}$	(i + j + k) / 3.0	Агар $i, j,$ ва k бутун сонлар бўлса, 3.0 махражидан фойдаланиб, сузувчи нуқтали бўлинмага таъсир қилади.

4.3 Ma’lumotlarni kiritish va chiqarish

4.3.1 Kiritish

Bu bo’limda siz o’zgaruvchiga foydalanuvchining ma’lumotini qanday joylashtirishni ko’rasiz. Dastur foydalanuvchining ma’lumotinii so’raganida, u avvalam bor foydalanuvchiga qaysi ma’lumotni kiritish kutilayotganini aytadigan xabarni chop etishi kerak.Bunday xabar prompt deb ataladi.

```
cout << "Please enter the number of bottles: "; // Display prompt
```

endl ni promptdan so’ng qo’shmang. Siz ma’lumotni keyingi qatorda emas, ikki nuqtadan so’ng paydo bo’lishini istaysiz.

So’ng, dastur ma’lumotni o’qiydigan buyruqni chiqaradi. Obyekt cin konsol oynasidan ma’lumot ni o’qiydi.Siz >> amalidan (ba’zida chiqarib tashlash amali deb ataladi) o’zgaruvchiga kiritish qiymatini joylashtirish uchun foydalanasiz. Bu usulda:

```
int bottles;
cin >> bottles;
```

Dastur kiritish operatorini amalga oshirganida,u foydalanuvchi kiritishni ta’minlashini kutadi.Bundan tashqari, foydalanuvchi Enter tugmasini dastur kiritishni qabul qilishi uchun bosishi kerak. Foydalanuvchi ma’lumotni kiritganidan so’ng,raqam bottles o’zgaruvchisiga joylashadi, va dastur davom etadi.E’tibor bering, bu kod segmentida shisha idishlar o’zgaruvchisini inisializasiya qilishga hojat yo’q, chunki u keyingi aynan o’sha segment tomonidan to’ldiriladi. Qoidaga ko’ra, siz o’zgaruvchini uni e’lon qilgandan so’ng inisializasiya qilishingiz kerak, agar u darhol kuzatilsa.

Siz bitta kiritish operatorida bir necha qiymatni o’qishingiz mumkin:

```
cout << "Please enter the number of bottles and cans: ";
cin >> bottles >> cans;
```

4.3.2 Formatlangan natijalar

Hisoblashning natijasini chop etganinigizda, siz uning ko’rinishini nazorat qilishni istaysiz. Masalan, siz yig’indini dollar va sentlarda chop etganingizda, siz odatda uni ikkita raqamga yaxlitlashni istaysiz.Mana bu siz xoxlagan natijning ko’rinishi

```
Price per ounce: 0.0409722 miqdorining o’rniga
Price per ounce: 0.04
```

Quyidagi dastur cout ga barcha suzuvchi nuqtali sonlar uchun o’nlik nuqtasidan so’ng ikkita raqamni ishlatishga ko’rsatma beradi:

```
cout << fixed << setprecision(2);
```

Bu buyruq hyech qanday natija bermaydi; u faqatgina coutni shunday ishlat adiki,u format natijasini o’zg art iradi. Belgilangan va aniqlangan qiymatlar

manipulyatorlar deb ataladi. Biz ularni 8 BOBda batafsil muhokama qilamiz. Hozircha, valyuta qiymatlari aniq ko'rinishini hohlasangiz, yuqorida berilgan operatorni kiritishni unutmang.

Manipulyatordan foydlanish uchun, siz dasturingizga <iomanip> header ni kiritishingiz shart:

```
#include <iomanip>
```

Siz manipulyatorlarni va qiymatlarni bitta operatorda nomoyon bo'lishi uchun birlashtirishingiz mumkin.

```
cout << fixed << setprecision(2)
<< "Price per ounce: "
<< price_per_ounce << endl;
```

Ba'zida qo'l keladigan yana boshqa manipulyator mavjud.Siz ma'lulomotlarning bir qancha qatorini namoyish etayotganingizda, siz odatda ustunlarni to'g'irlashni hohlaysiz. Siz setw manipulyatoridan keyingi chiqarish maydonining

kengligini o'rnatish uchun foydalanasiz. Kenglik bu qiymat, raqamlar , o'nlik nuqtalari va bo'shliqlarni ko'rsatish uchun ishlatiladigan belgilarning umumiy sonidir.Agar siz raqamlar ustunini bir qator tekis bo'lib kelishini istasangiz, kenglikni nazorat qilish muhimdir.

4.4 Masalaning yechimi: dastlab uni qo'llar yordamida bajarish

Algoritmni yaratishda eng muhim qadam bu dastlab hisoblashni qo'lda bajarishdir.Agar yechimni o'zingiz hisoblay olmasangiz, hisoblashni avtomatlashtiradigan dastur yoza olishingiz ehtimoldan uzoq.Qo'lda xisoblashdan foydalanishni namoyish etish uchun quyidagi masalani ko'rib chiqami. Oq va qora kafellarning qatori devor uzunasiga bo'ylab joylashtirilishi kerak.Estetika jihatdan me'mor birinchi va oxirgi kafel qora rangda bo'lish keraklini aniqladi.Sizning vazifangiz zarur bo'ladigan kafellarning sonini, oxirida qoladigan bo'sh joyni, mavjud oraliqni va har qaysi kafelning enini hisoblashdir.

4.5 Satrlar

Ko’p dasturlar raqamlar bilan emas matn bilan ishlaydi. Matn harflar, sonlar, tinish belgilari, bo’sh joylar kabi belgilardan tashkil topadi. Satr bu belgilarning ketma- ketligidir.Masalan, "Harry" satri beshta belgining ketma- ketligidan iboratdir.

4.5.1 Satrning turi

Satrlarni o’z ichiga olgan o’zgaruvchilarni aniqlashingiz mumkin.

```
string name = "Harry";
```

Ushbu satr turi C++ standartining qismi hisoblanadi.Undan foydalanish uchun oddiygina fayl nimini kiriting:

```
<string>:
```

```
#include <string>
```

Biz o’zgaruvchilar qatorini (yuqorida aniqlangan o’zgaruvchining nomi kabi) va doimiylarning satri ("Harry" kabi qo’shtirnoqqa olingan belgilarning ketma- ketligi) o’rtasidagi farqni aniqlymiz.O’zgaruvchilar satrida saqlangan satr o’zgarishi mumkin. Doimiy son (masalan 2) maxsus sonni ifoda etkanidek, satrning doimiysi ham maxsus satrni ifodalaydi.

Siz dastlabki qiymatni taqdim etmagan taqdiringizda ham, sonli o’zgaruvchilardan farqli ravishda, satrlar o’zgaruvchilarining inisializasiyalanishi kafolatlangan. Operatorsiz, Po umolchaniyu satr o’zgaruvchisi bo’sh ya’ni hyech qanday belgisiz satrga o’rnatiladi. Bo’sh satr doimiysi "deb yoziladi. Ta’rif

```
string response;
```

`string response = "";` kabi bir hil ta’sirga ega

4.5.2 Birlashtirish

Ikkita "Harry" va "Morgan" kabi satr berilgan.Siz ularni bitta uzun satrga birlashtirishingiz mumkin.Birinchi satrda natija barcha belgilardan iborat bo’lib, ikkinchi satrda barcha belgilar uning ketidan keladi. C++da siz + operatori

yordamida ikkita satrni birlashtirasiz.Masalan,

```
string fname = "Harry"; string lname = "Morgan";
string name = fname + lname;
```

satrda natija chiqadi

```
"HarryMorgan"
```

4.5.3 Kiritish satri

Siz satrni konsoldan o’qishingiz mumkin:

```
cout << "Please enter your name: ";
string name;
cin >> name;
```

Satr >> operatori bilan o’qilganda, faqatgina bitta so’z o’zgaruvchining qatoriga joylashtiriladi.Masalan, deylik o’zgaruvchining turlari

```
Harry Morgan
```

So’rovga javob.

```
Harry Morgan
```

4.5.4 Satrlarning vazifalari

Satrdagi belgilarning soni satrning uzunligi deb ataladi.Masalan,"Harry"ning uzunligi 5ga teng.Siz satrning uzunligini length funksiyasi yordamida hisoblashingiz mumkin. sqrt va pow funksiyalaridan farqli ravishda, length funksiyasi nuqtali qayd orqali ishga tushiriladi.Ya’ni, siz hohlagan uzunlikdagi satrni, keyin davrni,keyin funksiyaning nomini va so’ng qavslarni yozasiz:

```
int n = name.length();
```

Ko’p C++ dasturlari nuqtali notasiyani (qaydni) ishlatishni talab etadi, va siz qaysinisini qilish kerak yoki qilmaslik kerakligini eslab qolishingiz yoki topishingiz kerak.Ushbu funksiyalar komponentlik funksiyalari deb

ataladi.Komponentlik funksiyaning uzunligi o’zgaruvchining nomiga chaqiriladi deb aytamiz.

Satrni yozib bo’lgandan so’ng siz substr komponentlik funksiyasi yordamida satr ostini chiqarib tashlashingiz mumkin. Komponentlik funksiyasi chaqiruvi

```
s.substr(start, length)
```

returns a string that is made from the characters in the string s, starting at character start, and containing length characters. Here is an example:

```
string greeting = "Hello, World!";

string sub = greeting.substr(0, 5);

// sub is "Hello"
```

Mavzuga doir test savollari:

1. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){
 cout<<”Hello world”
}
```

a) Hello world
b) Kopilyatsi bo’ladi
c) Gramatik xatolik
d) Fayl topildi

2. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){
 int a, b=2; cin>>a>>b; b=3;
int c=a+b; cout<<c+b<<endl;
}
```

a) Kiritiladigan a ning qiymatiga bog’liq
b) Kiritiladigan a va b ning qiymatiga bog’liq
c) 6
d) 5

3. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){
 int a=1;  a++; cout<<a<<” ”;
```

```
int c=2; ++c; cout<<c <<endl;
}
```

a) 1 3
b) 2 2
c) 2 3
d) 3 2

4. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int a=12;
int main(){
 int a=1;  a+=1; cout<<a<<" ";
int c=2; --c; cout<<c <<endl;
}
```

a) 2 1
b) 2 2
c) 2 3
d) 3 2

5. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){
 int r=12, j=3; cout<<r%j<<" ";
int c=3; ++c; cout<<c%2 <<endl;
}
```

e) 0 0
f) 2 2
g) 2 3
h) 3 2

Nazorat savollari:

1. O’zgaruvchi nima?
2. O’zgaruvchining qanday turlari bor?
3. Operator nima va uning vazifasi nimadan iborat?
4. O’zlashtirish operatorining vazifasi nimadan iborat?
5. Izohlar nima uchun ishlatiladi?
6. Arifmetik amallar tartibi qandey bajariladi?

7. Inkrement va decrement amallari nima vazifani bajaradi?
8. Qoldiq olish amali qandey masalalarda ishlatiladi?
9. Kiritish va chiqarish operatorlarini ishlash usullarini bilasizmi?
10. Satrlar qandey o’zgaruvchilar?
11. Satrlarni turlari qandey bo’ladi?
12. Satrlarni kiritish operatorlari qandey ishlaydi?
13. Satrlarni chiqarish operatori qandey ishlaydi?

4.6. Operatorlar

4.6.1 if shart operatori

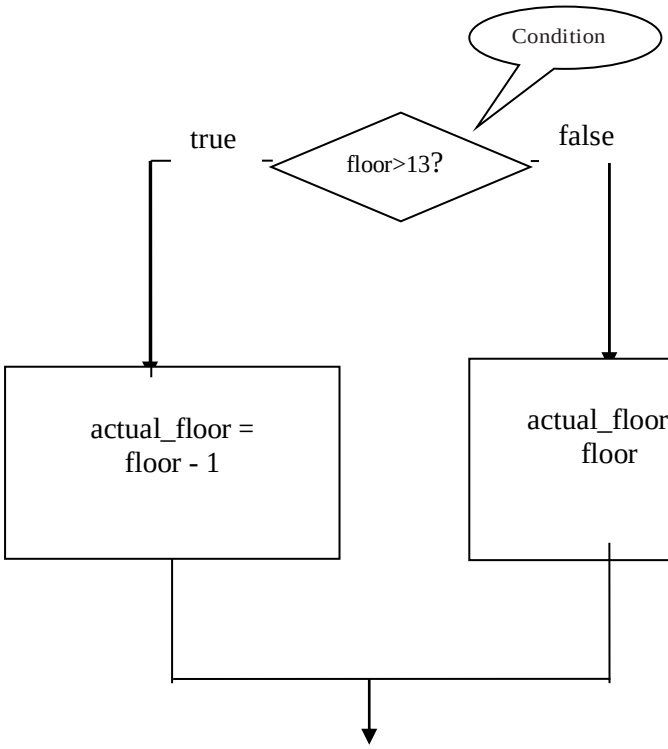
Shartli operator qarorni amalga oshirish uchun ishlatiladi. Qachonki sharoit bajarilsa bir gurux operatorlar amalga oshiriladi.Aks xolda, boshqa gurux operatorlar amalga oshiriladi.

Bu yerda shartli operatorning ishlatilishiga misol berilgan. Ko’pchilik mamlakatlarda 13 raqami omadsiz raqam xisoblanadi. Irimga ishonuvchilarni ko’ngliga g’ulg’ula solgandan ko’ra bino egalari bazida 13 qavatni qoldirib o’tishadi yani, 12 qavatdan so’ng birdaniga 14 qavat keladi. Albatta, 13 qavat odatda bo’sh qoldirilmaydi, bazi konspirasiya nazariyachilarining fikricha maxfiy idoralar va izlanishlar olib boriluvchi laboratoriyalar bilan band etilgan. U oddiygina 14 qavat deb ataladi. Bino liftlarini nazorat qiluvchi kompyuter bu tartibsizlikni o’rnini qoplashi zarur va 13 qavat yuqorisidagi barcha qavat raqamlarini o’zgartirishi kerak.

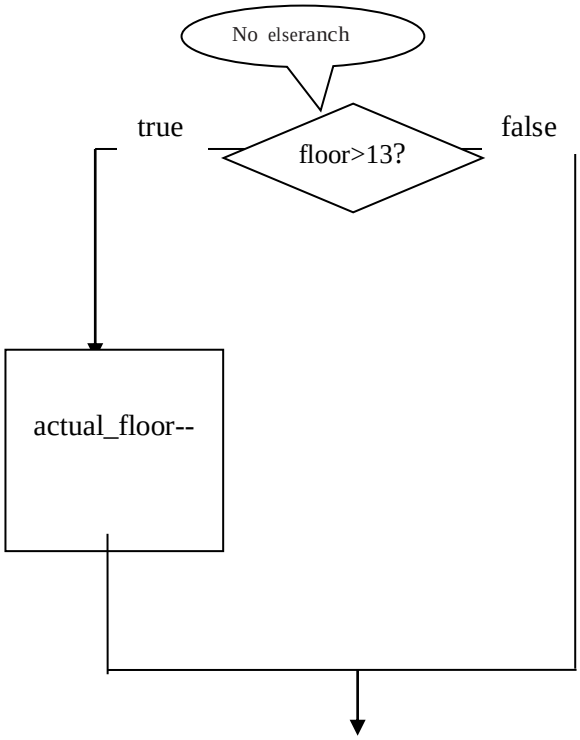
Keling bu jarayonni C++ da tasvirlaymiz. Biz foydalanuvchidan kerakli qavat raqamini yozishini so’raymiz va shundan so’ng xaqiqiy qavatni xisoblaymiz. Qachonki kiritilgan raqam 13 dan yuqori bo’lsa, kerakli qavatga yetishish uchun biz kiritilgan malumotni kamaytirishimiz kerak. Masalan, agarda foydalanuvchi 20 raqamini kiritsa, dastur 19 qavatni xaqiqiy qavat sifatida belgilaydi. Aks xolda , biz oddiygina yetkazib berilgan qavat raqamidan foydalanamiz.

```
int actual_floor;
if (floor > 13)
{
actual_floor = floor - 1;
 }
else
{
```

```
actual_floor = floor; }
```



3.1-rasm. shartli operator uchun diagramma



3.2-rasm. tarmoqsiz shartli operator uchun diagramma

Operatorning else tarmog'ida xech narsa qilib bo'lmaydi. Bunday xolatda, siz uni xuddi quyudagi misoldagi kabi to'liqligicha tushirib qoldirishingiz mumkin:

```
int actual_floor = floor;
if (floor > 13)
{
actual_floor--;

}
```

Quyudagi dastur shartli operatorni ishlashini yo'lga qo'yadi. Dastur kerakli qavatni so'raydi va so'ngra xaqiqatdagi qavatni chop etadi.

```
#include <iostream>
 using namespace std;
    int main()
    {
int floor;
cout << "Floor: ";
cin >> floor;
int actual_floor;
if (floor > 13)
{
actual_floor = floor - 1;
}
else
{
actual_floor = floor;
}
cout << "The elevator will travel to the actual
floor"
<< actual_floor << endl;
return 0;
}
```

Dasturni ishlatish

Floor: 20

The elevator will travel to the actual floor 19

Doimo tasmalardan foydalaning

Qachonki shartli operatorning asosiy qismi yakka operatordan tashkil topgan bo’lsa siz tasmalarni ishlatishingiz shart emas.

Masalan, quyudagi qonuniydir:

```
if (floor > 13)
floor--;
```

Shunday bo’lsada, xar doim tasmalarni ishlatgan maqul:

```
if (floor > 13)
{ floor--;
}
```

Sizning kodingizni o’qishni osonlashtiradi va sizning "odatiy xatolar" da ko’rsatilganidek xato qilish yextimolingizni kamaytiradi.

Jamlanmalarda nusxa ko’chirishdan ehtiyot bo’ling

Har bir jamlanmada koddan nusxa olayapsizmi yo’qmi bilish uchun qarang. Agar shunday bo’lsa, uni shartli operatordan tashqariga ko’chiring. Quyida shunday nusxa ko’chirishga misol berilgan:

```
if (floor > 13)
{
actual_floor = floor - 1;
cout << "Actual floor: " << actual_floor << endl;
}
else
{
actual_floor = floor;
cout << "Actual floor: " << actual_floor << endl;
}
```

Chiqarish operatori ikki jamlanmada ham bir hil. Bu xato emas, dastur to’g’ri ishlaydi. Shunday bo’lishiga qaramay, nusxa olingan operatorni surish orqali dasturni quyidagicha soddalashtirish mumkin:

```
if (floor > 13)
{
actual_floor = floor - 1;
}
Else
{
actual_floor = floor;
}
cout << "Actual floor: " << actual_floor << endl;
```

Dasturlar uzoq vaqtdan beri saqlanib turgan bo’lsa nusxani olib tashlash qisman muhim. Qachonki bir hil tasirli ikki to’plam operatorlar bo’lsa, dasturchi osongina faqat birsini o’zgartiradi.

4.6.2 Raqamlar va qatorlarni qiyoslash

Har bir shartli operator shartdan tashkil topgan. Ko’p hollarda, shart ikki qiymatni qiyoslashni taqozo etadi. Masalan, O’tgan misollarda biz quyidagi masalani ko’rdik: floor > 13. Qiyoslash > qiyosiy operator deyiladi. C++ da oltita qiyosiy operatorlar bor. (1jadvalga qarang).

Ko’rib turganingizdek, faqatgina ikki C++ qiyosiy operatorlari (> va <) huddi matematik belgi kabi ko’rinishga ega.

Kompyuter klaviyaturasida ≥, ≤, yoki ≠ lar uchun tugmalar yo’q , lekin >=, <=, va != operatorlar eslab qolishga oson, chunki ular bir biriga o’hshaydi. == operatori C++ ni endi o’rganayotganlar uchun boshida chalkashroq ko’rinadi. C++ da, = operatorining allaqachon manosi bor, yani vazifa (assignment).

C++	Matematik belgi	tarif
>	>	Greater than
>=	≥	Greater than or equal
<	<	Less than
<=	≤	Less than or equal
==	=	Equal
!=	≠	Not equal

“ == “ operatori tenglikni bildiradi:

```
floor = 13;

if (floor == 13)
```

Test orasida == operatorini va tashqarisida = operatorini ishlatishni esingizdan chiqarmasligingiz kerak

Shuningdek siz qatorlarni ham qiyoslashingiz mumkin:

```
if (input == "Quit")
```

Ikki qator turli xilligini tekshirish uchun != ni ishlating. C++ da xarf registri ahamiyatga ega. Masalan, "Quit"va "quit" bir qatorda emas.

Qiyosiy operatorga misollar - 3 <= 4 t o’ g’ r i 3 kamroq 4 dan; <= “nisbatan kamroq yoki teng” ligini 3 =< 4 Hato The “nisbatan kamroq yoki teng” operatori <=, =

bunday emas

Ифодалаш	Қиймат	Изоҳ
3 <= 4	тўғри	3камроқ 4 дан; <= "нисбатан камроқ ёки тенг" лигини текширади.
3 =< 4	**Ҳато**	The "нисбатан камроқ ёки тенг" оператори <=, =< бундай эмас. "нисбатан камроқ" белгиси биринчи келади.
3 > 4	нотўғри	> бу оператор <= операторнинг қарама қаршиси.
4 < 4	нотўғри	чап томон ўнг томондан кичикроқ бўлиши керак
4 <= 4	тўғри	икки томон ҳам тенг; <= "нисбатан камроқ ёки тенг"ни текширади
3 == 5 - 2	тўғри	==тенгликни текширади.
3 != 5 - 1	тўғри	!= нотенгликни текширади.3 5-1 эмаслиги тўғри.
3 = 6 / 2	**Ҳато**	== тенгликни текшириш учун фойдалан.
1.0 / 3.0 == 0.333333333	нотўғри	Қийматлар бир бирига жуда яқин бўлишига қарамасдан, улар аниқ тенг эмас. 86 саҳифадаги одатий ҳато 3.3 га қаранг

2- jadval C++ qiyosiy operatorni qanday ishlatilishini umumlashtirilgan

Quyidagi qaysi operatorlar to'g'ri, berilgan a bu 3 va bu 4?

```
a.  a + 1 <= b
```

```
b.  a + 1 >= b
```

```
c.  a + 1 != b
```

Qarqma-qarshi operator bering

```
floor > 13
```

Bu misolda qanday xato bor?

```
if (score_a = score_b)
{
cout << "Tie" << endl;
```

}

Bu shartli operatorda foydalanuvchi Y ga kirgan yoki yo’qligini tekshirish uchun shart yaratadi.

```
string input;
cout << "Enter Y to quit." << endl;
cin >> input;
if (...)
{
cout << "Goodbye." << endl;
return 0;
}
```

Qator bo’sh qator str emasligini qanday tekshirasiz ?

Nol ogohlantirish bilan tuzish

Tuzuvchi sizga ikki xil habar beradi: hatolar va ogohlantirishlar. Hato xabarlar falokatli; tuzuvchi dasturni bir yoki ko’proq hato bilan o’tkazmaydi.Ogohlantirish habarlari maslahatli; tuzuvchi dasturni o’tkazadi, lekin dastur siz kutganchalik ishlamasligiga olib keladi.

Ogohlantirishlarni sizning tuzuvchingiz bilan qanday qilib faollashtirishni o’rganish yaxshi g’oya, va boshqa ogohlantirishlar chiqmasligi uchun kod yozish kerak.Masalan, qyidagi testga qarang:

```
if (floor = 13)
```

Bir C++ tuzilmasi qiziqarli ogohlantirish beradi: “O’zlashtiruv to’g’ri qiymatda ishlatilsa atrofiga qavs qo’yish maslahat beriladi”. Shunisi achinarliki, habar yolg’on,chunki u talabalar uchun yozilmagan. Shunday bo’lishiga qaramasdan, bunday ogohlantirish sizni noto’g’ri operatorga e’tibor qaratishga va tuzatishga chorlaydi, bunday holatda = belgi == belgiga o’zgartiriladi.

Ogohlantirish yanada ko’zga tashlanadigan bo’lishi uchun, ko’pchilik

tuzuvchilar sizdan maxsus amallar bajarilishni talab qiladi. Bu integrasiyalashgan muhitdagi tanlov tugmasini bosishni yoki buyruq satridan maxsus tanlovni amalga oshirishni taqazo etadi. Rahbaringiz yoki labaratoriya yordamchisidan ogohlantirishlarni qanday qilib tuzuvchiga aylantirishni so'rang.

4.6.3 Ko'p sonli muqobillar

Siz shartli operator bilan ikki tomonlama jamlanmani qanday dasturlashni ko'rdingiz. Ko'p hollarda, ikkidan ko'p xolatlar mavjud.Bu bo'limda, qarorni qanday qilib ko'p sonli muqobillar bilan bajarishni ko'rasiz. Masalan, Rixter shkalasi bilan o'lchangan, zilzilani tasirini ko'rsatuvchi dasturga e'tibor qiling.

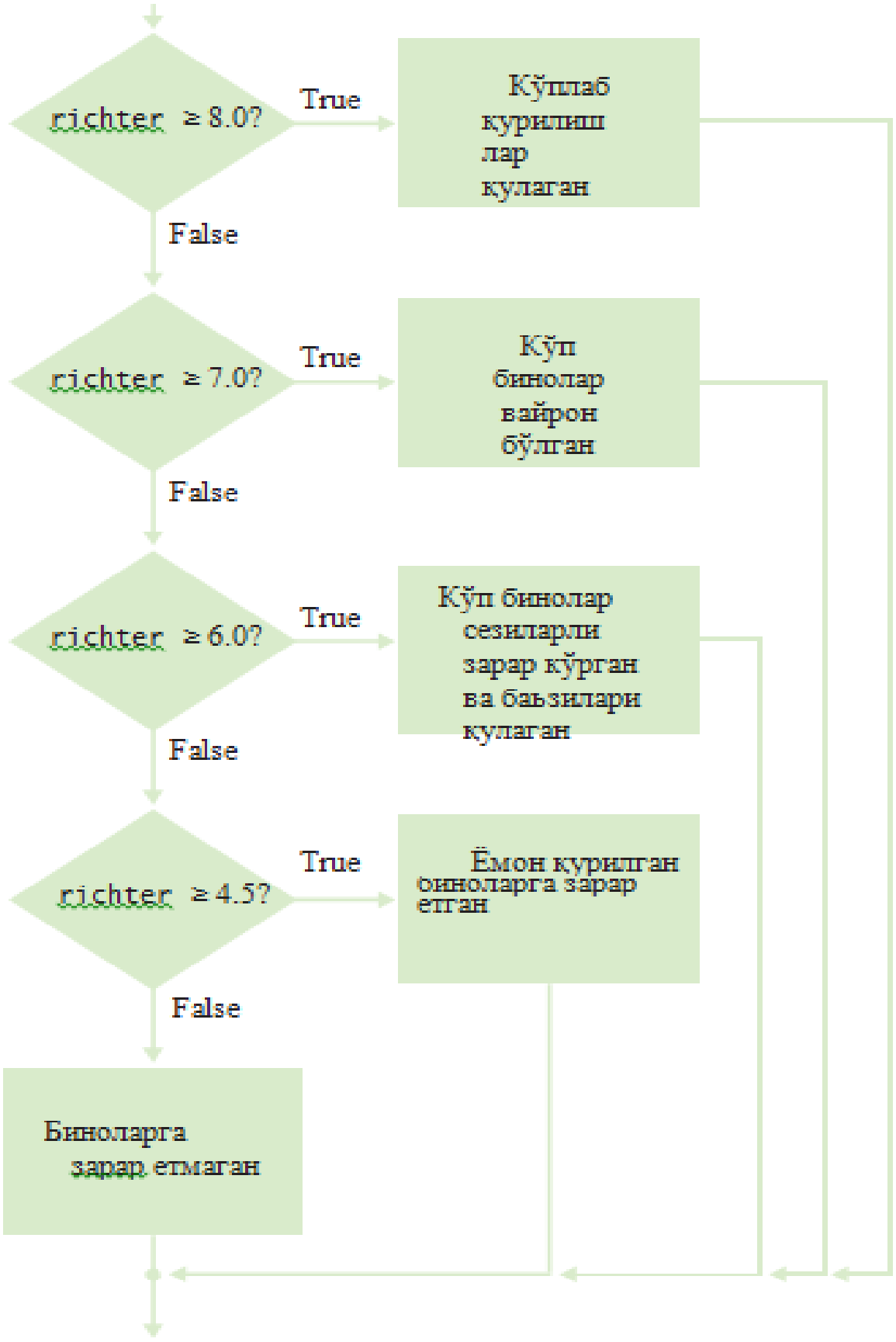
Rixter shkalasi zilzila kuchliligining o'lchov birligidir. Shkalaning har bir bosqichi, masalan 6.0 dan 7.0 ga, zilzila kuchini o'n baravar ortishini ko'rsatadi. bu holda, 5 ta bo'lim bor: zararning to'rtta tasvirining har biriga bittadan va zararlanmaganlik uchun ham bitta.

Ko'p sonli muqobillarni amalga oshirish uchun ko'p sonli shartli operatordan foydalan:

```
if (richter >= 8.0){
cout << "Most structures fall";
}
else if (richter >= 7.0){
cout << "Many buildings destroyed";
}
else if (richter >= 6.0){
cout << "Many buildings considerably damaged, some
collapse";
}
else if (richter >= 4.5){
cout << "Damage to poorly constructed buildings";
}
else {
cout << "No destruction of buildings";}
```

To'rtta testdan biri muvafaqqiyatli amalga oshirilishi bilan,tasir namoyish

qilinadiva boshqa tekshirishlar amalga oshirilmaydi.Agar to’rtta holatdan xech biri qo’llanilmasa, yakuniy else qism qo’llaniladi, va standart xabar chop etiladi.



Bu yerda siz operatorlarni saralashingiz kerak va birinchi bo’lib eng katta kesilishga qarshi tekshiring. Bizlar testlar tartibini teskari deb qaraylik:

```
if (richter >= 4.5) // Testlar noto’g’ri tartibda
{
cout << "Damage to poorly constructed buildings";
}
else if (richter >= 6.0)
{
cout << "Many buildings considerably damaged, some collapse";
}
else if (richter >= 7.0)
{
cout << "Many buildings destroyed";
]
else if (richter >= 8.0)
{
cout << "Most structures fall";
}
```

Bu ish bermaydi. Rixterning qiymati 7.1 deb taxmin qiling. U qiymat kamida

4.5, birinchi holatga to’g’ri keladi. Boshqa testlarga hyech qachon xarakat qilinmaydi.

Bu misolda, nafaqat ko’p sonli mustaqil shartli operatorlar, balki, shartli qismda else ning ketma ketligini ishlatishimiz ham muxim rol o’ynaydi. Mustaqil testlarning ketma ketligini e’tiborga oling:

```
if (richter >= 8.0)
{
cout << "Most structures fall";
}
if (richter >= 7.0)
{
cout << "Many buildings destroyed";
}
if (richter >= 6.0)
{
cout << "Many buildings considerably damaged, some collapse";
}
if (richter >= 4.5)
{
cout << "Damage to poorly constructed buildings";
}
```

Endi muqobillar o’ziga xos emas. Agar Rixter 7.1 bo’lsa, oxirgi uchta testlar bir biriga mos keladi, va uchta habar chop etiladi.

Tanlash operatori

Yolg’iz butun son qiymatini bir nechta doyimiy muqobillariga qiyoslaydigan shartli operatorlar ketma ketligi tanlash operator deyiladi. Masalan,

```
int digit;
switch (digit)
{
```

```
case 1: digit_name = "one"; break;
case 2: digit_name = "two"; break;
case 3: digit_name = "three"; break;
case 4: digit_name = "four"; break;
case 5: digit_name = "five"; break;
case 6: digit_name = "six"; break;
case 7: digit_name = "seven"; break;
case 8: digit_name = "eight"; break;
case 9: digit_name = "nine"; break;
default: digit_name = ""; break;
}
```

Buni quyidagicha ifodalashimiz ham mumkin:

```
int digit;
if (digit == 1) { digit_name = "one"; }
else if (digit == 2) { digit_name = "two"; }
else if (digit == 3) { digit_name = "three"; }
else if (digit == 4) { digit_name = "four"; }
else if (digit == 5) { digit_name = "five"; }
else if (digit == 6) { digit_name = "six"; }
else if (digit == 7) { digit_name = "seven"; }
else if (digit == 8) { digit_name = "eight"; }
else if (digit == 9) { digit_name = "nine"; }
else { digit_name = ""; }
```

Yaxshi, bu unchalik ham yorliq emas, lekin uni bir afzalligi bor. Ko’rinib

turibdiki, barcha jamlanmalar *same* qiymatini tekshiradi, yani *digit.*

Jamlanma uchun ko’p sonli (holat) *case* qismiga ega bo’lish mumkin, quyidagicha:

```
case 1: case 3: case 5: case 7: case 9:
odd = true; break;
```

Agarda hyech qaysi case qismlar bir biriga to’g’ri kelmasa standart jamlanma tanlanadi. Tanlashning ning xar bir jamlanmasi break qo’llanma bilan tugatilishi kerak . Agar break tushirib qoldirilsa, boshqaruv pastga keyingi jamlanmaga o’tadi, va oxirgi break ga yoki switch ning oxiriga yetmaguncha shunday davom etadi. Amalda, bu takrorlashdan chiqish xulqi unchalik foydali emas, lekin bu xatolarning oddiy sababi. Agar siz tasodifan break operatorni esdan chiqarsangiz, sizning dasturingiz tuziladi lekin kutilmagan kod bajariladi. ko’pchilik dasturchilar switch operatorni qandaydir havfli deb xisoblashadi va shartli operatorni ishlatishni avzal ko’rishadi.

Sizning kodingizda switch operatorni ishlatish yoki ishlatmaslikni biz sizga qo’yib beramiz. Qanday darajada bo’lsa ham, boshqa dasturchilarning kodida uchratsangiz sizda switch ni o’qiy olish qobilyati. bo’lishi kerak.

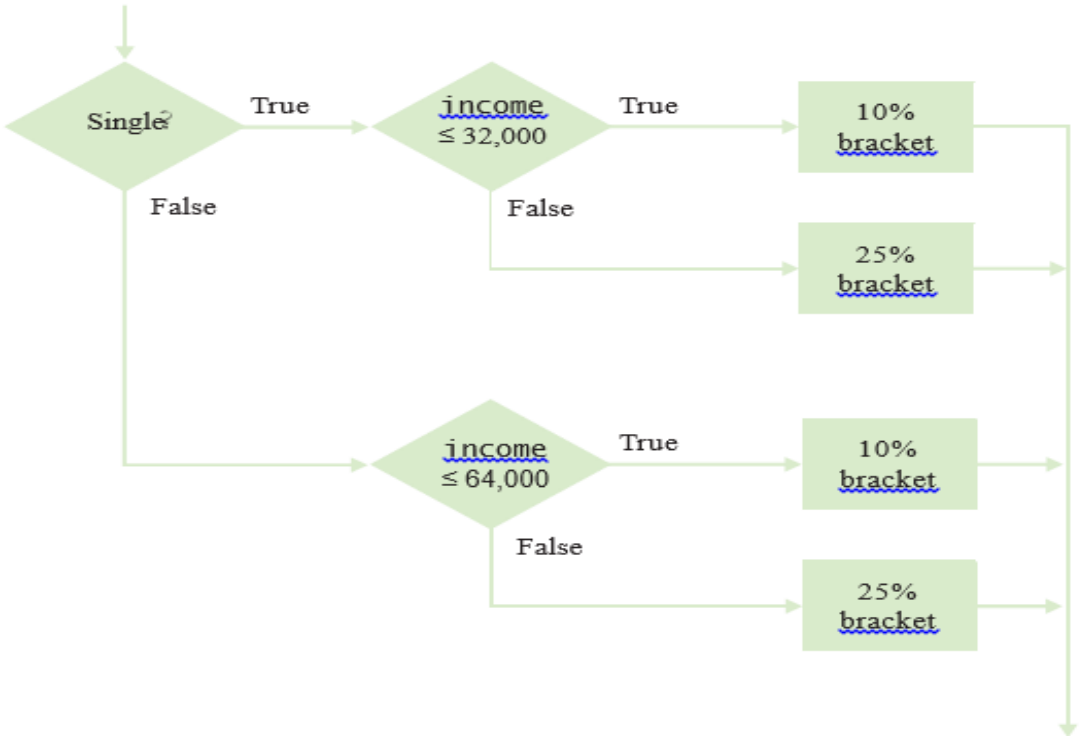
4.6.4. Ichma ich shartlar

Ko’pincha shartli operatorni boshqasini ichiga qo’shish zarur bo’ladi. Bunday sozlama ichma ich shartlar to’plami deyiladi. Bu yerda odatiy misol berilgan. Amerika qo’shma shtatlarida soliq to’lovchilarning oilaviy ahvoliga qarab turli xil soliq stavkalari ishlatiladi. Yolg’iz va turmush qurgan soliq to’lovchilar uchun farqli to’lov jadvali mavjud. Turmush qurgan soliq to’lovchilar o’zlarining daromadlarini qo’shib umumiy soliq to’laydilar.

2008 yilda amal qilgan jadvalni soddalashtirilgan shaklidan foydalanib, soliq stqvkasi xisoblarini beradi. Har bir "qatorga" turli xil soliq stavkalari amal qiladi. Bu jadvalda, birinchi qatorni daromadi 10 foiz soliqqa tortilgan va ikkinchi qatorni darromadi 20 foiz. har bir qatordagi daromad oilaviy ahvolga qarab chegaralanadi.

Xozir berilgan soliq to’lovchining statusiga va daromadi miqdoriga ko’ra

soliqlarni xisoblang. Asosiy tomoni shundaki, bu yerda xulosa qilishning ikki darajasi mavjud. Birinchisi, siz oilaviy axvolini tarmoqlashingiz kerak. Keyin, har bir to'lovchining statusini, daromad darajasiga qarab boshqa tarmoqlarga ajratishingiz kerak.



3.04-rasm. Daromad solig'ini xisoblash

Nazariyada, joylashtirish ikki qatlamdan ham chuqurroq bulishi mumkin. Ikki qatlam xulosa jarayoni (birinchi holatiga qarab, keyin oilaviy axvoliga qarab va so'ngra daromad darajasiga qarab) uchta ichiga olgan qatlamni talab qiladi.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
const double RATE1 = 0.10;
const double RATE2 = 0.25;
const double RATE1_SINGLE_LIMIT = 32000;
const double RATE1_MARRIED_LIMIT = 64000;
double tax1 = 0;
```

```
    double tax2 = 0;
    double income;
    cout << "Please enter your income: ";
    cin >> income;
cout << "Please enter s for single, m for married:";
    string marital_status;
    cin >> marital_status;
    }
    if (marital_status == "s")
    {
    if (income <= RATE1_SINGLE_LIMIT)
    {
    tax1 = RATE1 * income;
    }
    else
    }
    {
    tax1 = RATE1 * RATE1_SINGLE_LIMIT;
    tax2 = RATE2 * (income - RATE1_SINGLE_LIMIT);
    }
    }
    Else
    {
    if (income <= RATE1_MARRIED_LIMIT)
    {
    tax1 = RATE1 * income;
    }
    Else
    {
    tax1 = RATE1 * RATE1_MARRIED_LIMIT;
    tax2 = RATE2 * (income - RATE1_MARRIED_LIMIT);
    }
    }
```

```
double total_tax = tax1 + tax2;
cout << "The tax is $" << total_tax << endl;
return 0;
}
```

Natija:

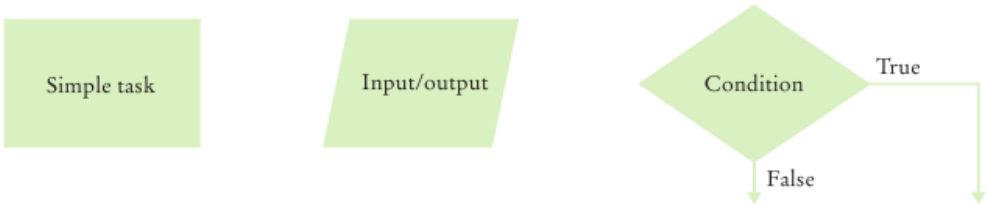
```
Please enter your income: 80000
Please enter s for single, m for married:
The tax is $10400
```

4.6.5 Muammoni hal etish: Diagrammalar

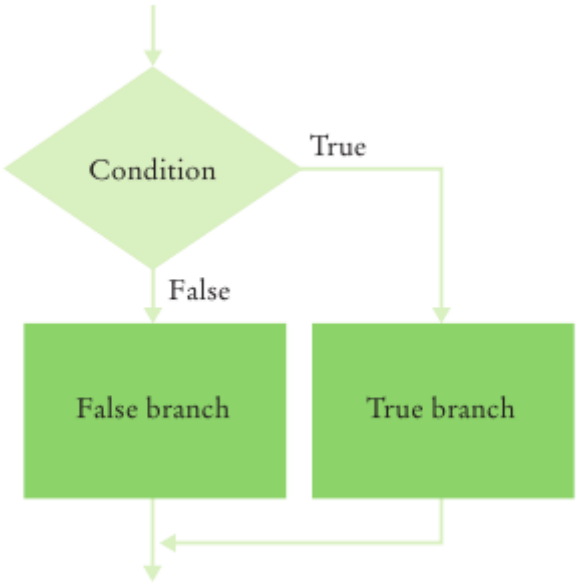
Siz shu bobning boshida diagrammalarga misollarni ko'rdingiz. Diagrammalar muammoni hal etish uchun kerak bo'ladigan xulosalar va vazifalarni tuzilishini ko'rsatadi. Qachonki siz murakkab muammoni hal etishingiz kerak bo'lsa,nazorat oqimini ko'z oldingizda gavdalantirish uchun diagramma chizganingiz yaxshi.

Diagrammaning asosiy elementlari 3.05-rasmda ko'rsatilgan.

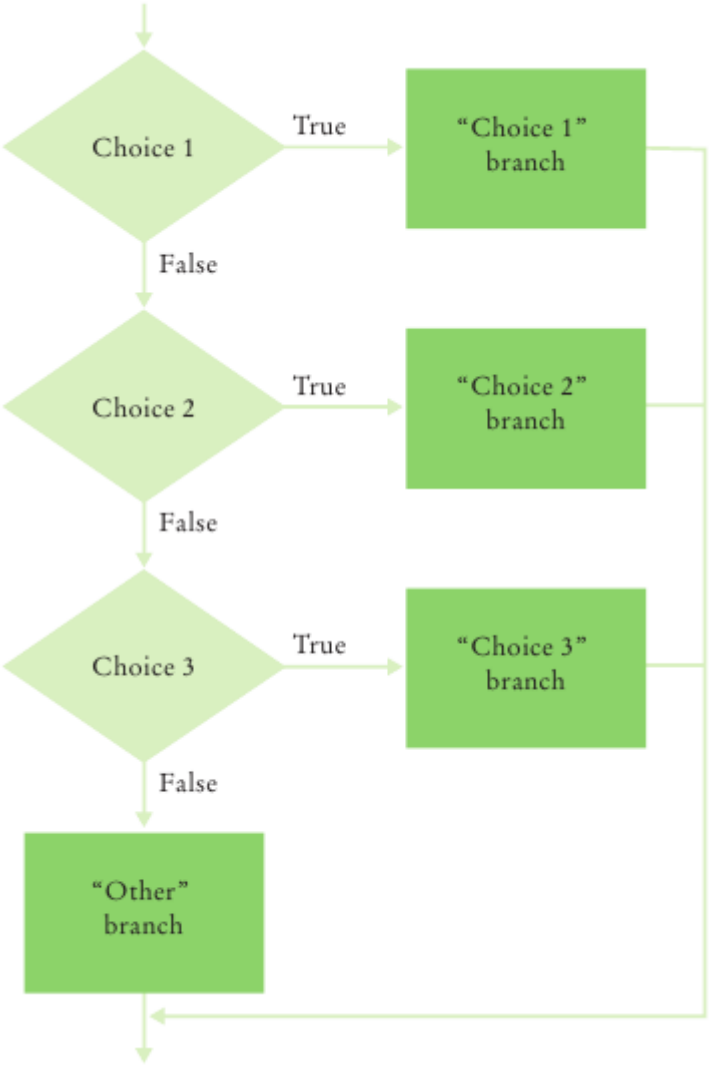
Asosiy g'oya yetarlicha sodda. Vazifalar kiritish\chiqarish bo'limlarini amalga oshirish keak bo'lgan tartibda birlashtiring. Qachon xulosa qilishingiz kerak bo'lsa, ikki natijali romb chizing. (3.06-rasmga qarang).



3.05-rasm. Diagramma elementlari



3.06-rasm. Ikki natijali diagramma



3.07-rasm. Ko'p sonli tanlovli diagramma

Har bir jamlanma vazifalar ketma ketligini va xattoki qo’shimcha xulosalarni o’z ichiga olishi mumkin. Agar qiymat uchun ko’p sonli tanlov bo’lsa, ularni 3.07-rasmdagi kabi tashqariga joylashtiring.

4.6.6 Bul o’zgarishlari va operatorlar

Bazida, siz siz dasturning bir qismida mantiqiy operatorni xisoblashingiz kerak va uni har joyda ishlating. To’g’ri yoki xato bo’lishi mumkin bo’lgan operatorni saqlash uchun Bul o’zgarishidan foydalanasiz. Bul o’zgarishlari mantiq ilmining asoschisi, matematik Jorj Bul nomi sharafiga qo’yilgan.

C++ da, Bul raqamlarini Buliin usulida yozishni bildiradi. Bulincha yozish o’zgarishlari aniq ikki qiymatda bo’ladi yani xato va to’g’ri. Bu qiymatlar qatorlar yoki butun sonlar emas;ular faqatgina bul o’zgarishlari uchun maxsus qiymatlar.

Testning holatida && operatori bilan qo’shilgan ikki qismi bor. (5 jadvalda S ilovada ko’rsatilganidek, the > va < operatorlarida && operatoriga qaraganda yuqori ustunlik bor.) Xar bir qismi to’g’ri yoki xato bo’lishi mumkin bo’lgan Bul qiymatidir. Agar ikki ifoda ham to’g’ri bo’lsa qo’shilgan ifoda to’g’ridir.Agar ikki ifodolardan biri xato bo’lsa,natija ham xato bo’ladi. (rasmga qarang).

a	B	a && B
true	true	true
true	false	false
false	true	false
false	false	false

a	B	a || B
true	true	true
true	false	true
false	true	true
false	false	false

a	!a
true	false
false	true

3.08-rasm. Bulning harakat jadvali

Aksincha, keling berilgan haroratda suvni suyuqlik bo’lmagan holatini tekshiramiz. Bu xaroratni 0 dan yuqoriroq va 100 dan pastroq holati. Ifodalarni qo’shish uchun || (yoki) operatorini ishlating:

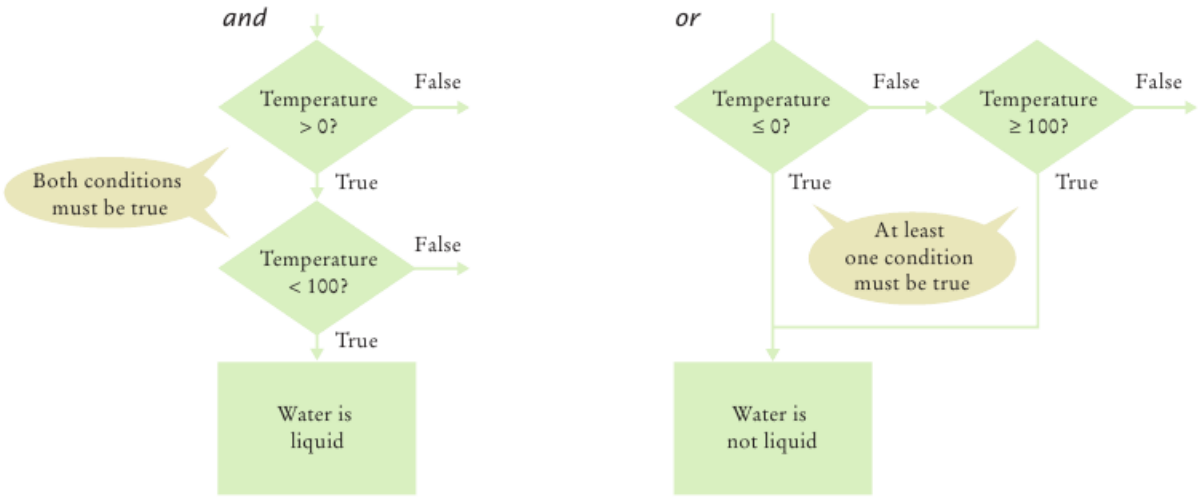
```
if (temp <= 0 || temp >= 100) { cout << "Not liquid"; }
```

3.09-rasmda bu misollarga diagrammalar ko’rsatilgan.

Bazida siz operatorni matiqiy bo’lmagan operator bilan almashtirishingiz kerak bo’ladi.! operatori yolg’iz holatda bo’ladi va qachonki shart xato bo’lsa to’g’ri deb xisoblanadi va aksincha agar shart to’g’ri bo’lsa xato deb xisoblanadi

Bu misolda agar muz xolatda bo’lmasa natijaga ega bo’lamiz.

```
if (!frozen)
{
cout << "Not frozen";
}
```



3.09-rasm. and va or kombinatsiyalari uchun diagramma

De Morgan qonuni

Insonlar uchun and/or ifodalarga murojaat qilmasdan mantiqiy vaziyatlarni tushunish qiyindir. Mantiqchi Augustus de Morgan nomiga qo’yilgan (1806-1871) De Morgan qonuni, Bulin ifodalarini soddalashtirish uchun ishlatiladi.

Faraz qiling, agar Qo’shma shtatlar kontinenti bo’ylab yuk tashimasak, eng katta kemada yuk tashish stavkasini hisoblashni hohlaymiz.

```
if (!(country == "USA"
&& state != "AK"
&& state != "HI"))
```

```
shipping_charge = 20.00;
```

Bu test bir oz murakkab va siz mantiqan o'ylashingizga to'g'i keladi. Agar shtat Gavayi bo'lmasa, shtat Alyaska bo'lmasa va mamlakat AQSh ekanligi to'g'ri emas, va narx 20.00.$. Ba'zi odamlar bu kod bilan adashmagan bo'lardi, deyish noto'g'ri. Kompyuter uchun farqi yo'q, lekin bu Dasturchilardan kod yaratishni va yozishni talab qiladi. Shuning uchun bunday vaziyatlarni qanday qilib soddalashtirish kerakligini bilish foydalidir.

De Morgan qonunini ikki shakli mavjud: birinchisi and ifodasini inkor etish uchun, ikkinchisi or ifodasini inkor etish uchun:

`!( A && B )` va **`!A || !B`** bir hil

`!( A || B )` va **`!A && !B`** bir hil

and va or operatorlari ichki harakatlanmasdan o'zgarishi faktligiga ahamiyat bering.

Masalan, "davlat Alyaska yoki Gavayi" ekanligini inkor qilish,

```
!(state == "AK" || state == "HI")
```

is "the state is not Alaska and it is not Hawaii":

```
!(state == "AK") && !(state == "HI")
```

Бу, Яъни қуйидагича

```
state != "AK" && state != "HI"
```

Кемада юк ташиш харажатини ҳисоблаш учун қонунга мурожаат қилинг:

```
!(country == "USA"
&& state != "AK"
&& state != "HI")
```

бунга тенг

```
!(country == "USA")
```

```
|| !(state != "AK")

|| !(state != "HI")
```

Ундан осонроқ тест келиб чиқади

```
country != "USA"

|| state == "AK"

|| state == "HI
```

and ва *or* ифодалар инкори билан вазиятни соддалаштириш учун, одатда, expressions, Де Морган қонунига мурожаат қилиб, инкорлик даражасини янада ошириш мумкин.

4.6.7 Amaliy dastur: Ma’lumot to’g’riligini tekshirish

Takrorlash operatori uchun muhim amaliyot bu ma’lumot to’g’riligini tekshirishdir. Dasturngiz foydalanuvchi ma’lumotini qabul qilgan zahoti, hisobni chiqarishdan avval qiymatlar to’g’riligiga ishonch hosil qilishingish kerak.

Lift programmamizga qarang. Aytaylik, lift panelida 1 dan 20 gacha tugmalar bor (13 yo’q). Quyidagilar noqonuniy raqamlar:

•13 soni

•nol yoki manfiy son

•20 dan katta son

•Raqamlar ketma-ketligida bo’lmagan son

Har bir holatda biz, xato haqida ma’lumot beramiz va programmani yopamiz.

Bu 13 raqamini oldini olish uchun:

```
if (floor == 13){

cout<<"Error: Thereis no thirteenth floor."<< endl;

return 1;}
```

Zudlik bilan asosiy funksiyani to’xtatadi va programmani tugallaydi. Agar programma yaxshi bajarilgan bo’lsa 0 qiymat bilan tugaydi, agar xato

hisoblanmagan bo’lsa noldan boshqa qiymat bo’ladi.

Mavzuga doir test savollari:

1. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){
int a=12; if(a>=1){a--;} else {a++;}
cout<<a<<endl;
}
```

a) 11
b) 13
c) 10
d) 14

2. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){
char s=’A’; int b=12;
if(s!=’A’ && b==12){cout<<”rost”<<endl;}
else {cout<<”Yolg’on”<<endl;}
}
```

a) rost
b) yolg’on
c) xatolik bor
d) xato ishlaydi

3. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){
char s=’A’; int b=12;
if(s!=’A’ || b!=12){cout<<”rost”<<endl;}
else {cout<<”Yolg’on”<<endl;}
}
```

e) rost
f) yolg’on
g) xatolik bor
h) xato ishlaydi

Nazorat savollari:

1. If operatorining vazifasi nima?

2. If operatorida qandey amallarni bajarish mumkin va qandey amallarni bajarish mumkin emas?
3. If operatorida {} ning vazifasi nimadan iborat?
4. If operatorida () ning vazifasi nimadan iborat?
5. If operatorida bool tipidagi o’zgaruvchilari nima uchun ko’p qo’llaniladi?
6. If operatorining qndey avzalliklarini bilasiz?

5. Takrorlanishlar

5.1 Davriy takrorlanishlar

Bu bo’limda siz maqsadga erishguncha takrorlanishni qanday takroran bajarilishini o’rganasiz.

1-bobdagi pul qo’yish muammosini eslang. Siz bank hisobingizga yiliga 5 foiz foyda keltiradigan 10,000$ qo’ydingiz. Byudjet balansi boshlang’ich qo’yilgan puldan ikki barobarga ko’payishi uchun necha yil kerak bo’ladi?

1-bobda biz bu muammoni yechish uchun quyidagi algoritmni tuzgandik.

yillik qiymat 0, keyingi ustun foyda, va 10,000 $ ga teng balans

year	interest	balance
0		$10,000

Yillik qiymatga 1 qo’shing.

Foydani balansni 0.05 ga ko’paytirib hisoblang(v.h.z. 5 foiz foyda).

Foydani balansga qo’shing.

So’nggi yil qiymati javob bo’ladi.

Endi siz C++ dagi parametrlarni qanday qilib aniqlash va yangilashni bilasiz. Siz xali" Daromad 20,000 $ ga yetmaguncha harakatni takrorlayverish" ni amalga oshirishni bilmaysiz.

Zarralar tezlatkichida atom ichidagi zarralar takrorlanish shaklidagi tunnelni bir necha marta kesib o’tadi va fizikaviy tajriba uchun talab qilinadigan tezlikka yetadi. Shunga o’xshab, kompyuter fanida takrorlanishda formulirovka holati aniq bo’lganda bajariladi.

C++ da, takrorlanish operatori quyidagi takrorni amalga oshiradi.

Kod

```
takrorlanish (vaziyat)
{
```

```
operatori
}
```

formulirovkani vaziyat to'g'irlanmaguncha bajarishda davom etaveradi. Bizning vaziyatda, biz yil ko'rsatkichini o'stirishni va balans mo'ljaldagi 20,000$ ga yetmaguncha foydani qo'shishni hohlaymiz.

```
takrorlanish (balans < moljal)
{
yil ++;
ikki baravar foyda = balans * miqdor / 100;
balans= balans + foyda;
}
```

Davomiy takrorlanish operatori takrorlanishga bir misoldir. Agar diagramma chizsangiz, takrorlanishlarning bajarilish diagramma chizig'i yana vaziyat tekshiriladigan nuqtaga boradi.

Misol uchun,ushbu takrorlanishdagi foyda miqdorini ko'rib chiqing: Qachon siz takrorlanishning asosiy qismi ichida o'zgarish aniqlasangiz, o'zgarish takrorlanishning har bir takrorlanishida sodir bo'ladi va har bir qaytarilish so'nggida yo'q bo'ladi. Quyidagi takrorlanishning o'zgarish qismini ko'rib chiqing.

```
takrorlanish (balans < moljal)
{
yil ++;
ikki barobar foyda = balans * miqdor / 100;
balans = balans+ foyda;
} // foyda bu yerda boshqa aniqlanmaydi
```

Aksincha, balans va yillik miqdorlar takrorlanish asosiy qismining

tashqarisida aniqlanadi. Shu yo’l bilan xuddi shunday miqdorlar takrorlanishning hamma takrorlanishida ishlatiladi.

Quyida sarmoya muammosini yechib beradigan dastur berilgan.

```
#include <iostream>
using namespace std;
int main()
{
    const double miqdor=5;
    const double=boshlangich_mablag=10000;
    const double moljal=2*boshlangich_mablag;
    ikki barobar mablag= boshlangich_mablag;
    int yil=0;
    takrorlanish(balans<moljal)
    {
        yil++;
        ikki_barobar_foyda=balans* miqdor/100;
        balans= balans+foyda;
}
hisob<<" Sarmoya … dan so’ng ikki barobar bo’ldi";
<<yil<<" yillar " <<yakun;
foyda 0;
}
```

Dasturni yuritish

Sarmoya 15 yildan so’ng 2 barobarga oshdi

5.2 Muammo yechimi: Belgilash

Dasturlash maslahatida, siz belgilash metodi haqida o’rgandingiz. Siz kod yoki belgili kodni belgilash (*Hand-Tracing*) qilsangiz, siz o’zgarishlar nomini bir varaq qog’ozga yozib olasiz, kodning har bir bosqichini xayolan bajarib o’zgarishlarni qaytadan yozasiz.

Kodni qog’ozga yozib olingani yaxshi. Elektr chizig’ini belgilab qo’yishga

marker, qog’oz, qisqichidan foydalaning. Qachon qiymat o’zgarishi bo’lsa, eski qiymatni o’chirib tagiga yangi qiymatni yozib qo’ying. Dastur natija chiqarsa, boshqa ustunga natijani yozib qo’ying.

Bu misolga qarang. Qanday qiymat ko’rsatilgan?

```
int n = 1729; int sum = 0; while (n > 0)
{
int digit = n % 10;
sum = sum + digit;
n = n / 10;
}
cout << sum << endl;
```

bu yerda 3 ta qiymat bor: n, sum va digit.

n sum digit

Birinchi ikki qiymat takrorlanish kiritilishidan avval 1729 va 0 bilan boshlangan.

```
int n = 1729;
int sum = 0;
 while (n > 0)
{
int digit = n % 10;
sum = sum + digit;
n = n / 10;
}
cout << sum << endl;
```

n noldan kattaroq bo’lgani uchun takrorlanish kiriting. O’zgaruvchan son 9 deb olinadi(1729 soni 10 ga bo’linganda qoladigan qoldiq). Miqdor yig’indisi 0 +

= 9 ga teng. O’zgaruvchan summa 0+9=9 deb olinadi.

```
int n = 1729;
int sum = 0;
 while (n > 0)
{
int digit = n % 10;
sum = sum + digit;
n = n / 10;
}
cout << sum << endl;
```

Nihoyat, n 172 ga aylandi. (1729 / 10 bo’linmadagi chiqarib tashlangan qoldiqni eslang,chunki sonlarning ikkisi ham butun sonlar edi.)

Eski qiymatni o’chirib tashlang va yangi qiymatni eskisi tagiga yozib qo’ying.

```
int n = 1729;
int sum = 0;
while (n > 0)
{
int digit = n % 10;
sum = sum + digit;
n = n / 10;
}
cout << sum << endl;
```

Endi yana takrorlanish holatini tekshiring .

```
int n = 1729;
```

```
int sum = 0;
 while (n > 0)
{
int digit = n % 10;
sum = sum + digit;
n = n / 10;
}

cout << sum << endl;
```

Chunki n haliyam noldan kattaroq, takrorlanishni qaytaring.Endi raqam 2 bo'ldi, summa 9+2=11 bo'ldi, va n 17 ga yetdi.

Bu yerda maqsadli takrorlanishni oddiy ishlatilishi. Biz bujadvalda ko'rsatilganidek, yillar davomida tejab kelgan foydamizni chiqaramiz:

YIL	BALANS
1	10500.00
2	11025.00
3	11576.25
4	12155.06
5	12762.82

Maqsadli takrorlanish qo'llaniladi chunki, o'zgaruvchi yil 1 da boshlanadi va bu mo'ljalga yetib borguncha doimiy ravishda o'sib boradi:

```
for (int year = 1; year <= nyears; year++)
    {
    Balansni yangila
    Balans va yilni chiqar.   }
```

5.3 Bajaruvchi takrorlanishlar

Ba’zida siz takrorlanishning asosiy qismini kamida bir marta bajarishni va asosiy qism bajarilganda takrorlanish testini o’tkazishni hohlaysi. Bajaruvchi takrorlanish quyidagi maqsadda ishlatiladi:

```
Bajaruvchi
{
operatorlar
}
while (vaziyat);
```

Bajaruvchi takrorlanishning asosiy qismi birinchi amalga oshiriladi, keyin vaziyat tekshiriladi. Ba’zi odamlar bunday takrorlanishni takroran tekshiruv takrorlanishi deb atashadi, chunki vaziyat takrorlanish asosis qismi tugatilgandan keyin test qilinadi. Aksincha, davriy va maqsadli takrorlanishlar testgacha bo’lgan takrorlanishlardir. Bunday takrorlanish turlarida, vaziyat takrorlanish asosiy qismiga kirgunga qadar test qilinadi.

Bunday takrorlanishga oddiy misol kiritilgan ma’lumotlarni isbot qilishdir. Faraz qiling, siz foydalanuvchidan < 100 qiymatni kiritishni so’radingiz.Agar foydalanuvchi ahamiyat bermay kattaroq qiymat kiritsa, siz qiymat to’g’ri bo’lmaguncha yana so’rang.Albatta, siz qiymatni foydalanuvchi kiritmaguncha test qilolmaysiz. Bu bajaruvchi takrorlanishga mos keladi (5-shaklga qarang):

```
boshlangich qiymat;
bajar
{
cout << "qiymat kiriting < 100: ";
cin >> value;
}
```

```
while (qiymat >= 100);
```

5.4 Ma’lumotlarni qayta ishlash

Bu bo’limda siz qayta qanday qilib kiritilga ma’lumotlar ketma-ketligini ko’rib chiqish va o’qishni o’rganasiz.

Ma’lumotlar ketma-ketligini o’qisangiz, sizga natija oxirini ko’rsatuvchi metod kerak bo’ladi. Ba’zida siz omadlisiz va hyech qanday qiymat nol bo’lmasligi mumkin. Shunda siz foydalanuvchiga sonlarni kiritishda davom etishini yoki 0 kiritib ketma-ketlikni tugatishni aytishingiz mumkin. Agar nol kiritish mumkin bo’lib, manfiy sonlar mumkin bo’lmasa siz tugallanganlikni ko’rsatish uchun 1 dan foydalanishingiz mumkin. Tugatish uchun signal vazifasini bajaruvchi qiymat signalizator deb ataladi.

Keling bu texnikani oylik qiymatini o’rtachasini hisoblaydigan dasturda ishlatib ko’ramiz. Namunaviy dasturimizda, biz 1 dan signalizator sifatida foydalanamiz. Ishchi albatta, kam oylikka ishlamasligi mumkin, lekin bepul ishlaydigan ko’ngillilar ham bo’lishi mumkin.

Sikl ichida biz kiritilgan ma’lumotni o’qiymiz. Agar kiritilgan son 1 bo’lmasa biz uni qayta ko’rib chiqamiz. O’rtachasini hisoblash uchun biz barcha oyliklarni umumiy yig’indisini .

```
while (...)
{
cin >> oylik;
if (oylik != -1)
{
summa = summa + oylik;
hisob ++;
}
}
```

Biz takrorlanishda signalli qiymat topilmaguncha to’xtab turamiz.

```
while (oylik != -1)
{
...
}
```

Bu yerda bir muammo bor: Sikl birinchi marta kiritilganda,ma’lumotlar qiymati o’qilmaydi. Oylik maoshni signalizatordan ko’ra bir oz qiymat bilan boshlang :

```
ikkilangan maosh = 0; // 1 dan boshqa hohlagan qiymat bajaradi
```

Qo’shimcha, bajaruvchi takrorlanishni ishlating

```
bajar
{
...
}
takrorlanish (oylik != -1)
```

Quyida berilgan dastur foydalanuvchi signalizatorni kiritguncha kiritilgan ma’lumotlarni o’qiydi va keyin o’rtachasini hisoblab chiqaradi.

```
#include <iostream>
using namespace std;
int main()
{
double sum = 0;
int count = 0;
double salary = 0;
cout << "Enter salaries, -1 to finish: ";
```

```
while (salary != -1)
{
cin >> salary;
if (salary != -1)
{
sum = sum + salary;
count++;
}
}
if (count > 0)
{
double average = sum / count;
cout << "Average salary: " << average << endl;
}
else
{
cout << "No data" << endl;
}
return 0;
}
```

Dasturni yuritish

```
Enter salaries, -1 to finish: 10 10 40 -1
Average salary: 20
```

Raqamli signalli belgilar faqatgina kiritilgan ma’lumotlarda cheklov

bo’lsa ish beradi. Ko’p hollarda,yo’q bo’lsa ham. Masalan,siz manfiy qiymat yoki 0 dan iborat ma’lumotlar to’plamini o’rtacha qiymatini hisoblamoqchisiz. Siz kiritilgan ma’lumot yakunini ko’rsatish uchun 0 yoki 1 dan foydalana olmaysiz. Bu vaziyatda ma’lumotlarni ular barbod bo’lguncha o’qishingiz mumkin. 3.8 bo’limda ko’rganingizdek, vaziyat

```
cin.fail()
```

agar ishlayotgan ma’lumotlar barbod bo’lsa to’g’ri. Masalan,kiritilgan ma’lumotlar bu operatorlar bilan o’qildi:

```
double value;
cin >> value;
```

Agar foydalanuvchi son bo’lmagan qiymatni kiritsa(Q kabi) keyin ma’lumotlar barbod bo’ladi.

Biz hozir qo’shimcha qiyinchilikka duch kelamiz. Siz faqatgina ma’lumotni uni o’qishga va takrorlanish kirgazishga harakatingizdan keyin muvaffaqiyatsizlikka uchraganini bilasiz.Muvaffaqiyatsizlikni esda tutish uchun Buliin o’zgaruvchanligidan foydalaning:

```
cout << "Enter values, Q to quit: ";
bool more = true;
while (more)
{
cin >> value;
if (cin.fail())
{
more = false;
}
else
```

```
{
Process value.
}
}
```

Ba’zi Dasturchilar takrorlanishni nazorat qilish uchun mantiqiy o’zgaruvchanligidan foydalanishni yoqtirmaydilar. Shunga qaramasdan, ma’lumot o’qishning oson yo’li bor.

Ifoda

```
cin >> value;
```

ҳолатда ишлатилиши мумкин. Бу агар cin қийматни ўқигандан кейин барбод бўлмаса бу тўғри деб баҳоланади. Шунинг учун сиз маълумотлар тўпламини қуйидаги цикл билан ўқишингиз ва қайта ишлашингиз мумкин:

```
while (cin >> value)
{
Process value.
}
```

Bu takrorlanish kiritilgan ma’lumotni bir ketma-ketligini qayta ishlash uchun qulay.

5.5 Oddiy takrorlanish algoritmlari

Bu bo’limda, takrorlanish deb amalga kiritilgan eng ko’p uchraydigan algoritmlarni ko’rib chiqamiz. Ulardan takrorlanish loyihangiz uchun boshlang’ich davri sifatida foydalanishingiz mumkin.

5.5.1 Yig’indi va o’rtacha qiymat

Kiritilgan ma’lumotlar soni yig’indisini hisoblash oddiy vazifa, o’rta yig’indini yozib qo’ying: har bir kiritilgan ma’lumot qiymatini qo’shasiz. Albatta, jami 0 bilan chiqishi kerak.

```
double total = 0;
double input;
while (cin >> input)
{
total = total + input;
}
```

O’rtachasini hisoblash uchun qancha qiymat borligini sanang va jamiga bo’ling. Jami 0 emasligiga ishonch hosil qiling.

```
double total = 0;
int count = 0;
double input;
while (cin >> input)
{
total = total + input;
count++;
}
double average = 0;
if (count > 0) { average = total / count; }
```

5.5.2 Sonlarning o’zgarishi

Ma’lum bir shartni bajarish uchun siz qancha qiymat kerakligini bilishni hohlaysiz.Bir qatorda qancha bo’sh joy borligini bilishni hohlaysiz. Qayerda o’zgarish bo’lsa ko’tarilgan va 0 bilan boshlanadigan hisoblagich yuriting.

```
int spaces = 0;
for (int i = 0; i < str.length(); i++)
{
```

```
string ch = str.substr(i, 1);
if (ch == " ")
{
spaces++;
}
}
```

Masalan, agar str qator "Oq ko’ng il xonimim" bo’lsa, (i 2 va 7 bo’lganda) intervallar ikki baravar ko’payadi. Intervallar o’zgarishi takrorlanish tashqarisida kuzatiladi. Biz takrorlanishni bir o’zgarishni yangilashini hohlaymiz ch o’zgarishi takrorlanish ichida kuzatiladi. Har bir iterasiya uchun alohida o’zgarish kiritiladi va har bir takrorlanish iterasiyasi so’nggida olib tashlanadi.

Bu takrorlanish kiritilgan ma’lumotlarni tezda ko’rib chiqish uchun ham ishlatiladi. Quyidagi takrorlanish bir vaqtda so’z, tekst o’qiydi va uch harfdan iborat so’zlarning sonini hisoblaydi:

```
int short_words = 0;
string input;
while (cin >> input)
{
if (input.length() <= 3)
{
short_words++;
}
}
```

5.5.3 Birinchi o’zgarishni topish

Shartni bajaradigan qiymatlarni hisoblasangiz, barcha qiymatlarni ko’rib chiqishingiz kerak.Biroq, agar vazifangiz o’zgarish topish bo’lsa, shart bajarilgan

zahoti to’xtashingiz mumkin.Bu yerda qatorda birinchi intervalni topadigan takrorlanish berilgan. Chunki qatordagi har bir elementni ko’rib chiqmaymiz, davriy takrorlanish maqsadli takrorlanishdan ko’ra yaxshi tanlov:

```
bool found = false;
int position = 0;
while (!found && position < str.length())
{
string ch = str.substr(position, 1);
if (ch == " ") { found = true; }
else
{
 position+;}
}
```

Agar o’zgarish topilsa,topilma to’g’ri bo’ladi va pozisiya birinchi o’zgarishning indeksi bo’ladi. Agar takrorlanish o’zgarishni topmasa takrorlanish yakunidan so’ng topilma noto’g’riligicha qoladi.

Avvalgi misolda biz qatorda vaziyatni o’zgartiradigan belgini o’rgandik. Xuddi shunday jarayonni foydalanuvchi kiritgan ma’lumot uchun qo’llashingiz mumkin. Aytaylik, foydalanuvchidan musbat qiymat< 100 kiritishni so’rayapsiz. Foydalanuvchi to’g’ri ma’lumot kiritguncha so’rayvering:

```
bool valid = false;
double input;
while (!valid)
{
cout << "Please enter a positive value < 100: ";
cin >> input;
```

```
if (0 < input && input < 100) { valid = true; }
else { cout << "Invalid input." << endl; }
```

O’zgaruvchan ma’lumot davriy takrorlanish tashqarisida kuzatiladi,chunki siz ma’lumotdan takrorlanish tamom bo’lganidan keyin foydalanmoqchisiz. Agar bu takrorlanish asosiy qismining ichida kuzatilganda bundan takrorlanish tashqarisida foydalana olmasdingiz.

5.5.4 Maksimum va minimum

Natijadagi eng katta qiymatni hisoblash uchun, siz duch kelgan eng katta elementlarni yig’ib kattarog’ini topganingizda yangilang.

```
double largest;
cin >> largest; double input;
while (cin >> input)
{
if (input > largest)
{
largest = input;
}
}
```

Bu algoritm kamida bitta ma’lumot kiritishni talab qiladi.

Eng kichik qiymatni hisoblash uchun taqqoslashni teskarisiga aylantiring:

```
double smallest;
cin >> smallest;
double input;
while (cin >> input)
{
```

```
if (input < smallest)
{
smallest = input;
}
}
```

5.5.5 Ketma-ket kelgan qiymatlarni taqqoslash

Sikldagi qiymatlarni ko’rib chiqayotganda, ba’zida qiymat undan oldingi qiymat bilan solishtirishingiz kerak bo’ladi.Masalan, siz 1 7 2 9 9 4 9 kabi kiritilgan ma’lumotlar ketma-ketligida ikkita bir xil son yonma-yon kelganmi yo’qmi tekshirmoqchisiz.

Qiyinchilikka duch keldingiz. Qiymatni o’qiydigan oddiy takrorlanishni ko’rib chiqing:

```
double input;
while (cin >> input)
{
// Kiritilgan ma’lumotlar shulardan iborat
…
}
```

Hozirga ma’lumotni oldingisi bilan qanday solishtirasiz? Har doim ma’lumotlar avvalgilari qaytadan yozilgan shu paytdagi ma’lumotlardan iborat bo’ladi.

Javob avvlgi ma’lumotlarni eslab qolish, bunday:

```
double input;
double previous;
while (cin >> input)
{
```

```
    if (input == previous) { cout << "Duplicate input"
<< endl; }
    previous = input;
    }
```

Bir muammo qoldi. Sikl birinchi marta kiritilganda, avvalgisi xali qo'yilmagan bo'ladi. Bu muammoni takrorlanish tashqarisida boshlang'ich ma'lumot operasiyasi bilan hal qilishingiz mumkin:

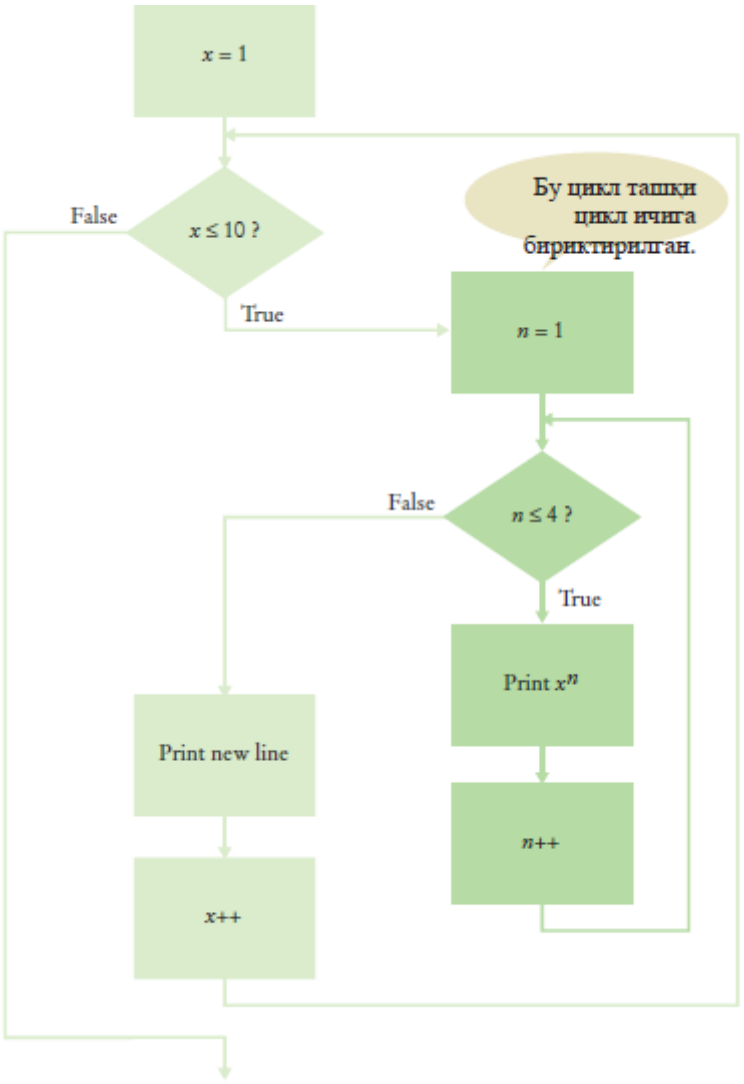
```
    double input;
    double previous;
    cin >> previous;
    while (cin >> input)
    {
    if (input == previous) { cout << "Duplicate input"
<< endl; }
    previous = input;
    }
```

5.6 Kiritilgan takrorlanishlar

Bo'limda, siz ikkita shartli operatorni qanday joylashtirishni ko'rdingiz. Shunga o'xshab, murakkab iterasiyalar ba'zida biriktirilgan takrorlanishni talab qiladi: bir takrorlanish boshqa takrorlanish ichida bo'ladi.Jadval tuzish paytida biriktirilgan takrorlanishlar o'zi paydo bo'ladi.Tashqi takrorlanish jadval qatorlari bo'ylab takrorlanadi.Ichki takrorlanish mavjud qator ustunlari bilan ishlaydi.

Jadval qatorini qanday chiqarasiz? Har bir daraja uchun qiymat chiqarishingiz kerak.

Bu takrorlanish avvalgi takrorlanish ichiga joylashtirilishi kerak. Ichki takrorlanish tashqi takrorlanish ichiga biriktirildi (shaklga qarang).



Tashqi takrorlanishda 10 ta qator bor.Har bir x uchun, programma ichki takrorlanishda 4 ta ustun chiqaradi. Demak, 10 × 4 = 40 qiymat chiqariladi.

5.7 Tasodifiy sonlar va modellashtirish

Modellashtirish programmasi haqiqiy hayotdagi biror faoliyatni modellashtirish uchun ishlatiladi (yoki tasavvurdagi). Modellashtirishdan ob-havoni oldindan aytish, yo'l-harakatini tahlil qilish, biznes va fanda ko'pgina amaliyotlar uchun foydalaniladi. Bu bo'limda, tasodif bilan modellashtirilgan modellarni amalda qo'llashni o'rganasiz.

5.7.1 Taxminiy sonlarni paydo qilish

Real dunyodagi ko'p hodisalarni mutlaqo aniq oldindan taxmin qilishimiz qiyin, ko'pincha o'rtacha holatini bilishimiz mumkin. Masalan, do'kon o'z tajribasidan xaridor har 5 minutda kelishini biladi. Albatta, bu o'rtacha olganda-

haridor har bir 5 minut interval oralig'ida kelmaydi. Xaridor yo'nalishini aniq modeli uchun taxminiy beqarorlikni hisobga olishingiz kerak bo'ladi.Bunday modellashtirishni kompyuterda qanday amalga oshirasiz?

C++ kutubxonasida umuman taxmin sonlarni ishlab chiqaradigan taxminiy sonlar generatori bor. Rantga murojaat qilib 0 bilan Rant maksimum oralig'idagi sonlarni chiqarish mumkin. Rantga yana murojaat qiling, endi boshqa sonlarni olasiz. Rand funksiyasi <cstdlib> asosida ko'rsatilgan.

Quyida berilgan programma rant funksiyasiga 10 marta murojaat qilgan.

```
int main()
{
for (int i = 1; i <= 10; i++)
{
int r = rand();
cout << r << endl;
}
return 0;
}
```

Program Run

```
1804289383
846930886
1681692777
1714636915
1957747793
424238335
719885386
1649760492
596516649
118964142
```

Aslida sonlar butunlay tasodifiy emas. Ular uzoq vaqt davomida qaytarilmagan sonlar natijasida yoziladi. Bu natijalar oddiy formulalar hisobidan kelib chiqqan;ular faqat tasodif sonlarga o'xshaydi.Shuning uchun ular ko'pincha soxta tasodifiy sonlar deb ataladi.

Dasturni yana yurgazishga harakat qiling.Yana xuddi shunaqa natija olasiz.

Bu tasodifiy sonlar formulalardan kelib chiqishini tasdiqlaydi.Biroq modellashtirish paytida aynan bir xil natija olishni istamaysiz.Bu muammoni hal qilish uchun tasodifiy sonlar natijasi uchun boshlang'ich sonni aniqlab oling. Har safar yangi boshlang'ich sondan foydalanganingizda tasodifiy sonlar generatori yangi natija chiqarishni boshlaydi. Boshlang'ich son rant funksiyasi bilan o'rnatiladi. Boshlang'ich son sifatida foydalaniladigan qiymat bu hozirgi vaqtdir:

```
srand(time(0));
```

Tasodifiy sonlarni chiqarishdan avval programmangizda bu murojaatni qilib ko'ring.

Har safar programma yuritilganda tasodifiy sonlar har xil bo'ladi.
Shuningdek, vaqtfunksiyasiniko'rsatadigan<vaqt> sarlavhasiniyamkirgazing.

5.7.2 Kubikli qur'a tashlashni modella

Talab bo'yicha, tasodifiy son generatoridagi natijani turli xil diapazonga ko'chirish kerak. Masalan, kubik otishni modellashtirish uchun sizga 1 va 6 o'rtasidagi taxminiy sonlar kerak.

Bu ikki chegara a va b o'rtasidagi taxminiy butun sonlarni hisoblash uchun oddiy ko'rsatma. 4.3-dasturlash maslahatidan ma'lumki, a va b o'rtasida ikki chegara bilan hisoblaganda b - a + 1 qiymatlar bor. Birinchi, 0 va b - a o'rtasidagi qiymatga ega bo'lish uchun % (b - a + 1) rantni hisobla, keyinani qo'sh va a va b o'rtasidagi taxminiy qiymatni chiqar:

```
int r = rand() % (b - a + 1) + a;
```

Mana bu programma 6 qirrali toshni otishni modellashtiradi.

```
#include <iostream>
```

```
#include <string>
#include <cstdlib>
#include <ctime>
using namespace std;
int main()
{
srand(time(0));
for (int i = 1; i <= 10; i++)
{
int d1 = rand() % 6 + 1;
int d2 = rand() % 6 + 1;
cout << d1 << " " << d2 << endl;
}
cout << endl;
return 0;
}
```

Program Run

```
5 1
2 1
1 2
5 1
1 2
6 4
4 4
6 1
6 3
5 2
```

Mavzuga doir test savollari:

1. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){
for(int i=1; i<=4; i++ ){cout<<i<<” ”;}
}
```

a) 1 2 3 4
b) 1 2 3 4 5
c) xatolik bor
d) 1 2 3

2. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){
for(int i=4; i<=1; i-- ){cout<<i<<” ”;}
}
```

a) 4 3 2 1
b) 4 3 2 1 5
c) xatolik bor
d) 3 2 1

3. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){
for(int i=1; i<=3; i+=2 ){cout<<i<<” ”;}
}
```

a) 1 3
b) 1 3 5
c) xatolik bor
d) 3 5

4. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){   int i=1;
for( ; i<=4; i++ ){cout<<i<<” ”;}
}
```

a) 1 2 3 4
b) 1 3 5
c) xatolik bor
d) 1 2 3 4 5

5. Quyidagi ifoda qanday natija chiqaradi?

```
#include<iostream>
int main(){   int i=1;
for(int j ; i<=10; i++ ){cout<<i<<" ";}
}
```

a) 1 2 3 4
b) 1 3 5
c) xatolik bor
d) 1 2 3 4 5

Nazorat savollari:

1. Dasturlashda takrorlanish jarayoni nimani anglatadi?
2. Takrorlanishni qandey operatorlar bilan tashkil qilinadi?
3. Qandey takrorlanish operatorlari mavjud?
4. Ichma ich takrorlanish jarayonlarini tashkil etish shartlari qandey?
5. Tasodifiy sonlarni keltirib chiqarish operatori nima va u qandey ishlaydi?
6. Takrorlash operatorlari yordamida qandey masalalarni yechish mumkin?
7. Kiruvchi va chiquvchi ma’lumotlarni takrorlanish operatorlari yordamida qayta ishash turlari bilasizmi?
8. Eng ko’p ishlatiladigan takrorlash operatori nima?
9. Takrorlash operatorlarining bir biridan avzallik tomonlari nimada?
10. Takrorlash operatorlarining har biri maxsus masalalar uchun mo’ljallangan bo’lsa? Nima uchun 1 masalani yechish uchun bir nechtasidan foydalnish kerak?

6. Funksiyalar

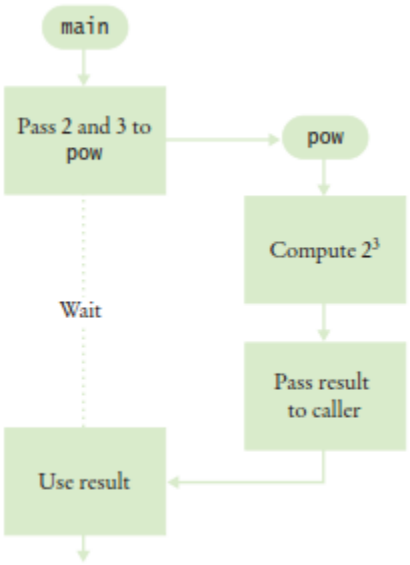
6.1. Funksiyalar qora quti(black boxes) sifatida

Funksiya bu- nomi mavjud bo’lgan ko’rsatmalar ketma ketligi. Siz allaqachon bir necha funksiyaga duch kelgansiz.Misol tariqasida 2- bobda keltirilgan power xy.ni xisoblashga doir ko’rsatmalarni o’z ichiga olgan "pow"deb nomlangan funksiyani keltirish mumkin. Bundan tashqari xar bir C++ dasturda main deb nomlangan funksiya mavjud.

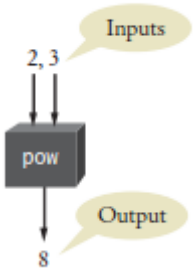
Ko’rsatmalarni amalga oshirish maqsadida siz funksiyaga murojaat qilasiz. Misol uchun quyidagi dasturga qarang:

```
int main()
{
double z = pow(2, 3);
...
}
```

pow(2, 3)ifodasini qo’llab, main pow funksiyasini power 23ni xisoblash uchun chaqiradi. main funksiyasi vaqtinchalik to’xtatiladi. pow funksiyasining ko’rsatmalari natijani amalga oshiradi va xisoblaydi . pow funksiyasi natijani qaytaradi (bu 8chi qiymat) yana main ga va main funksiyasi dastur amalini qayta boshlaydi. (1chi rasmga qarang).



5.01-Rasm Chaqiruv funksiyasi davomida dasturni amalga oshirish jarayoni



5.02-rasm. pow funksiyasi qora quti (black ox)sifatida

Boshqafunksiya pow funksiyani chaqirganda , u "chiqish" (input) ni ta’minlaydi , pow (2,3) chaqiruvidagi 2 chi va 3 chi ifodaga o’xshab . Bu ifodalar argumentlar deb ataladi.

Mazkur termin inson tomonidan ta’minlangan boshqa kiritishlar (input) bilan bo’ladigan chalkashliklarni oldini oladi. . Shunga o’xshab pow funksiyasi amalga oshiradigan chiqish (output) qaytish qiymati deb ataladi.

6.2. Funksiyalarni amalga oshirish

Ushbu bo’limda siz funksiyani berilgan detallar(spesifikasiya) asosida qay tarzda amalga oshirishni o’rganasiz.Biz juda oddiy misolni qo’llaymiz:tomonlar uzunligi berilgan kub hajmini aniqlash funksiyasi.

cube_volume funksiyasi kub hajmini aniqlash uchun berilgan tomonlar

uzunligidan foydalanadi.

Ushbu funksiyani yozayotganda, siz:

• Funksiya uchun nom tanlashingiz kerak.(cube_volume).

• Har bir argument uchun o'zgaruvchini (peremennaya) ko'rsatishingiz lozim. (`double side_length`). Bu o'zgaruvchi parametrlar o'zgaruvchisi deb ataladi.

• Qaytish qiymatining turini aniqlashingiz kerak (double).

Funksiya ta'rifining birinchi qatorini tuzish uchun quyidagi barcha ma'lumotlarni birga qo'ying:

```
double cube_volume(double side_length)
```

Co'ng funksiya chaqirilganda amalga oshiriladigan operator ya'ni funksiya body (tana) sini aniqlang.

Tomonlar uzunligi s bo'lgan kubning hajmi s × s × s. Biroq,aniqroq bo'lishi uchun bizning parametr o'zgaruvchimiz side_length deb atalgan , s emas, shu sababli biz side_ length * side_length * side_length ni aniqlashimiz lozim. Biz bu qiymatni volume deb nomlangan o'zgaruvchida saqlaymiz.

```
double volume = side_length * side_length *
side_length;
```

Funksiya natijasini qaytarish uchun qaytish operatoridan foydalaning:

return volume;

Funksiya tanasi(body) qavsda keltirilgan.Mana to'liq funksiya:

```
double cube_volume(double side_length)
{
double volume = side_length * side_length *
side_length;
return volume;
}
```

Keling bu funksiyani ishlatamiz. Biz cube_volume funksiyasini 2 marta chaqiradigan main funksiyasi bilan ta’minlaymiz.

```
int main()
{
double result1 = cube_volume(2);
double result2 = cube_volume(10);
cout << "A cube with side length 2 has volume " << result1 << endl;
cout << "A cube with side length 10 has volume " << result2 << endl;
return 0;
}
```

Qachonki funksiya turli xil argumentlar bilan chaqirilsa, funksiya turli xil natijalarni qaytaradi. cube_volume(2) chaqiruvini e’tiborga oling. 2chi argument side_ length parametr o’zgaruvchisiga mos keladi. Shu sababli bu chaqiruvda side_length bu 2. Funksiya side_length * side_length * side_length, yoki 2 * 2 * 2.ni amalga oshiradi. Qachonki funksiya turli xil argumentlar bilan chaqirilganda,aytaylik 10 bilan, so’ngra funksiya 10 * 10 * 10 ni aniqlaydi.

Endi biz ikkala funksiyani test dasturida birlashtiramiz. Chunki main calls cube_ volume ni chaqiradi . cube_volume funksiyasini main funksiyasini aniqlanishdan oldin bilish lozim. Bunga cube_volume ni dastlabki faylga birinchi va main ni esa oxirida joylashtirish orqali osongina erishiladi.

Mana to’liq dastur.Funksiya harakatini tasvirlaydigan izohga e’tibor bering.

```
#include <iostream>
using namespace std;
/**
Computes the volume of a cube.
```

```
    @param side_length the side length of the cube
    @return the volume
    */
    double cube_volume(double side_length)
    {
    double volume = side_length * side_length *
side_length;
    return volume;
    }
    int main()
    {
    double result1 = cube_volume(2);
    double result2 = cube_volume(10);
    cout << "A cube with side length 2 has volume " <<
result1 << endl;
    cout << "A cube with side length 10 has volume " <<
result2 << endl;
    return 0;
    }
```

6.3. Parametrni uzatish

Ushbu bo'limda biz funksiyalarga parametr uzatish mexanizmini ko'rib chiqamiz. Funksiya chaqirilganda ,parametr o'zgaruvchi hosil bo'ladi. (Parametr o'zgaruvchi o'rnida yana bir keng qo'llaniladigan atama bu rasmiy parametrdir.)Chaqiruv funksiyasida har bir parametr o'zgaruvchiga taqdim etiladigan ifoda bu argu ment deb ataladi. (Bu ifoda uchun qo'llaniladigan yana bir atama bu xaqiqiy parametrdir(actual parameter.) Har bir parametr o'zgaruvchi mos

keluvchi argument qiymati bilan ilk holatga keltiriladi.

cube_volume funksiyasining side_length parametr o’zgaruvchisi yaratildi.

• Parametr o’zgaruvchi chaqiruvda uzatilgan argument qiymati bilan ilk holatga keltirildi. Bizning holatda , side_length 2ga keltirildi. 2

• Funksiya 8 qiymatga ega side_length * side_length * side_length,

ifodasini hisoblaydi . Bu qiymat o’zgaruvchi volume da saqlanadi.3 3

•Funksiya qaytadi .Uning barcha o’zgaruvchilari olib tashlanadi. Qaytish qiymati chaqiruvchiga yuboriladi.

Endi keyingi cube_volume(10) chaqiruvida nima sodir bo’lishiga e’tibor bering.Yangi parametr o’zgaruvchi hosil bo’ldi. (Birinchi chaqiruv cube_volume ga qaytarilganda oldingi parametr o’zgaruvchi olib tashlanganligini eslang.) Bu 10 argument bilan ilk holatga qaytarildi va jarayon takrorlanadi.2- chaqiruv funksiyasi tugatilgandan so’ng uning o’zgaruvchilari yana olib tashlanadi.

Boshqa o’zgaruvchilardek parametr o’zgaruvchi faqatgina mos keluvchi turlar qiymatiga qo’yish mumkin.Misol uchun cube_volume funksiyasining side_length parametr o’zgaruvchisida ikkilangan format mavjud. U cube_volume(2.0) yoki cube_volume(2)ni chaqirish kuchiga ega.Keyingi chaqiruvda butun son 2 avtomatik tarzda ikkilangan 2.0.qiymatga aylanadi. Biroq cube_ volume("two") chaqiruvi qonuniy emas.

6.4 Qaytish qiymati

Qaytish operatoridan funksiya natijasini aniqlash uchun foydalanasiz.Qaytish operatori qayta ishlangandan so’ng funksiya darhol ishni tugatadi . Bu harakat boshidagi aloxida holatlarni boshqarish uchun qulay:

```
double cube_volume(double side_length)
{
if (side_length < 0) { return 0; }
double volume = side_length * side_length *
side_length;
```

```
return volume; }
```

Agar funksiya side_length uchun inkor qiymat bilan chaqirilsa ,funksiya 0 va qoldiqqa qaytadi.

Oldingi misolda har bir qaytish operatori konstanta yoki o'zgaruvchini qaytaradi. Haqiqatda qaytish operatori xoxlagan ifodaning qiymatini qaytarishi mumkin.Qaytish qiymatini o'zgaruvchida saqlash va o'zgaruvchini qaytarish o'rniga o'zgaruvchini bartaraf etib yanada murakkabroq ifoda qaytarish imkoniyati ko'proq:

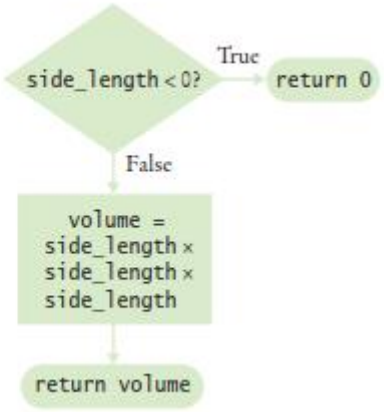
```
double cube_volume(double side_length)
{
return side_length * side_length * side_length;
}
```

Har bir funksiyaning qismi qiymat qaytarishi muhim xisoblanadi. Quyidagi noto'g'ri funksiyani e'tiborga oling:

```
double cube_volume(double side_length)
{
if (side_length >= 0)
{
return side_length * side_length * side_length;
} // Xato
}
```

Siz cube_volume ni inkor qiymat bilan chaqirishingizni taxmin qiling. Albatta uni chaqirishingiz taxmin qilinmaydi,biroq bu noto'g'ri kodlash natijasida yuz berishi mumkin.Chunki if holati to'g'ri emas ,qaytish operatori amalga oshirilmaydi. Biroq funksiya nimanidir qaytarishi lozim. Vaziyatga qarab tuzuvchi buni xato deb belgilashi mumkin,yoki funksiya tasodifiy qiymatni qaytarishi

mumkin.



Ba’zi xavfsiz qiymatni qaytarish orqali bu muammodan ximoyalanish mumkin:

```
double cube_volume(double side_length)
{
if (side_length >= 0)
{
return side_length * side_length * side_length;
}
return 0;
}
```

Har bir funksiyaning oxirgi operatori qaytish operatori bo’lishi kerak. Bu esa funksiya oxiriga yetganda ,ba’zi qiymatlar qaytishini ta’minlaydi.

6.4.1 Qaytish qiymatining yo’qligi

Funksiya doimo nimanidir qaytarishi kerak. Agar funksiya kodi bir necha bo’limni o’z ichiga olsa ,ularning har biri qiymat qaytarishiga ishonch hosil qiling:

```
int sign(double x)
{
if (x < 0) { qaytish -1; }
```

```
if (x > 0) { qaytish 1; }
// Xato: agar x 0 ga teng bo'lsa qaytish qiymati bo'lmaydi.
}
```

Bu funksiya raqam belgisini xisoblaydi: –1 manfiy sonlar uchun va +1 musbat sonlar uchun. Agar argument 0 bo'lsa hyech qanday qiymat qaytarilmaydi. Ko'pgina tuzuvchilar bu hollarda ogohlantirish berishadi, biroq siz ogohlantirishga e'tibor bermasangiz va funksiya 0 argument bilan chaqirilsa tasodifiy son qaytariladi.

6.4.2 Funksiya e'lonlari

Tuzuvchi bilmaydigan funksiyani chaqirishi bu tuzish paytidagi xato hisoblanadi, xuddi aniqlanmagan o'zgaruvchini ishlatishdagi xatoga o'xshab. Funksiyalarni ishlatishdan oldin ularni aniqlab olsangiz bu xatoni oldini olasiz.. Birinchi bo'lib dasturdagi quyi daraja yordamchi funksiyalarni,keyin o'rta darajadagi va nixoyat asosiy yordamchi funksiyalarni aniqlab oling.

Ba'zi dasturchilar o'z dasturlarida birinchi bo'lib asosiy funksiyalarni ro'yxatga olishni afzal ko'rishadi.Agar siz xam shuni afzal ko'rsangiz dastur tepasida boshqa funksiyalarni qanday qilib e'lon qilishni o'rganib olishingiz kerak. Funksiya e'loni qaytish turi ,funksiya nomi,va parametr o'zgaruvchini ro'yxatga oladi,biroq tanani (body) o'z ichiga olmaydi:

```
double cube_volume(double side_length);
```

Bu e'lon funksiya yana qayerdadir amalga oshirilishini bildiradi. E'lonni ta'rifdan ajratish oson.E'lonlar nuqtali vergul bilan tugaydi,ta'rif esa {...} blok bilan keladi. E'lonlar shuningdek prototip deb ham ataladi.

Funksiya prototipida parametrlar nomi majburiy emas.Siz yozishingiz mumkin:

```
double cube_volume(double);
```

Biroq har bir parametr maqsadini hujjatlash maqsadida parametr nomini

kiritish yaxshi xisoblanadi.

pow va boshqa matematik funksiyalar kabi oddiy funksiyalar e'loni ham sarlavhali fayllarda joylashgan bo'ladi.Agar siz <cmath> ning ichini ko'rsangiz, u yerda siz pow va boshqa matematik funksiyalar e'lonini topasiz.

Bu cube.cpp faylining muqobil tuzilishi:

```
#include <iostream>
using namespace std;
// cube_volume ning e'loni
double cube_volume(double side_length);
int main()
{
double result1 = cube_volume(2); // Use of cube_volume
double result2 = cube_volume(10);
cout << "A cube with side length 2 has volume " << result1 << endl;
cout << "A cube with side length 10 has volume " << result2 << endl;
return 0;
}
// cube_volume ning ta'rifi
double cube_volume(double side_length)
{
return side_length * side_length * side_length;
}
```

Agar sizga shu yo'nalish yoqsa,ikkilanmasdan bundan dasturingizda

foydalaning. Siz faqatgina bitta kamchiligini bilishingiz lozim.Agar siz funksiya nomini yoki parametr turini o’zgartirsangiz siz uni 2 joyda ,ham e’londa ham ta’rifda to’g’rilashingiz kerak.

6.5 Qaytish qiymatisiz funksiyalar

Ba’zan siz qiymat qaytarmaydigan ko’rsatmalar ketma ketligini amalga oshirishingizga to’g’ri keladi. Agar o’sha ko’rsatmalar ketma ketligi ko’p marotaba sodir bo’lsa ,siz uni funksiyaga joylashtirishni xoxlab qolasiz. C++ da,qaytish qiymatining yo’qligini ko’rsatish uchun void qaytish turidan foydalanasiz. Mana odatiy misol.Sizning vazifangiz xuddi mana bunga o’xshab oynada qatorni yozishdan iborat:

```
------
-
!Hello
! ----
---
```

Biroq turli xil qatorlar Hello ning o’rnida ishlatilishi mumkin.Bu vazifa uchun funksiya quyidagicha aniqlanadi:

```
void box_string(string str)
```

Vazifani yechish uchun umumiy metodni tuzgan holda siz endi funksiya tanasini rivojlantirasiz.

• - n harfini o’z ichiga olgan a chizig’ini chizing + 2 marta, n bu qator uzunligi

• Chap va o’ng tomondan a ! bilan o’ralgan qatorni o’z ichiga olgan a chiziq chizing.

• - n harfini o’z ichiga olgan yana boshqa chiziq chizing + 2 marta

Mana funksiyaning bajarilishi:

```
/**
```

Oynada qator chizing.

@param str yozish uchun qator

*/

```
void box_string(string str)
{
int n = str.length();
for (int i = 0; i < n + 2; i++) { cout << "-"; }
cout << endl;
cout << "!" << str << "!" << endl;
for (int i = 0; i < n + 2; i++) { cout << "-"; }
cout << endl;
}
```

Bu funksiya hyech qanday qiymatni xisoblamasligini e’tiborga oling.U ba’zi harakatlarni bajaradi va chaqiruvchiga uzatadi (O’xshash dasturda ko’ring ch05/box.cpp.)

Qaytish qiymati bo’lmagani uchun siz box_string ni ifodada qo’llay olmaysiz. Siz chaqirishingiz mumkin

```
box_string("Hello");
```

ammo

result = box_string("Hello"); // Error: box_string natija uzatmaydi.

natija Xato:

Agar siz oxiriga yetmasdan turib void funksiyasidan qaytmoqchi bo’lsangiz, siz qiymatsiz qaytish operatoridan foydalanasiz. Misol uchun,

```
void box_string(string str)
{
int n = str.length();
```

```
if (n == 0)
{
return; // Darhol qaytish
}
...
}
```

6.6. Muammoni yechish: Takroran ishlatiladigan funksiyalar

C++standart kutubxonasidan ko’p funksiyalarni ishlatgansiz. Bu funksiyalar dasturchilarga ularni qayta tiklashga ehtiyoj qolmasligi uchun standart C++ bir bo’lagi sifatida berilgan . Albatta C++ kutubxonasi har bir yuz berishi mumkin bo’lgan ehtiyojni qamrab olmaydi. Ko’p marotabali muammolarda qo’llaniladigan shaxsiy funksiyangizni yaratib siz vaqtingizni tejaysiz.

Aynan o’sha dastur yoki alohida dasturlarda deyarli o’sha kod yoki ramziy kod yozsangiz, funksiya ko’rsatishga e’tiborga oling.Mana bu kod nusxalashga odatiy misol:

```
int hours;
do
{
cout << " 0 va 23 orasidagi qiymatni kiriting: ";
cin >> hours;
}
while (hours < 0 || hours > 23);
int minutes;
do
{
```

```
cout << " 0 va 59 orasidagi qiymatni kiriting: ";
cin >> minutes;
}
while (minutes < 0 || minutes > 59);
```

Ushbu dastur segmenti har bir o’zgaruvchi aniq bir qator ichida ekanligiga ishonch hosil qilgan holda ,2 ta o’zgaruvchini o’qiydi. Oddiy harakatni funksiyaga chiqarib olish oson:

```
/**
Foydalanuvchiga asosli kiritish(vvod)ni ta’minlamaguncha berilgan maksimumgacha qiymat kiritishni aytib turadi.
@param high eng katta ruxsat beriladigan kiritish (vvod)
@return foydalanuvchi bergan qiymat ( 0 va high orasida, barchasini o’z ichiga oladi)
*/
int read_int_up_to(int high)
{
int input;
do
{
cout << " 0 va" << high << ": "orasidagi qiymatni kiriting;
cin >> input;
}
while (input < 0 || input > high);
return input;
}
```

So’ng bu funksiyani 2 marta qo’llang:

```
int hours = read_int_up_to(23);

int minutes = read_int_up_to(59)
```

Biz hozir takrorlashning nusxalashini o’chirdik,u faqat bir marta funksiya ichida sodir bo’ladi.

Funksiya butun son qiymatini o’qishi kerak bo’lgan boshqa dasturlarda takroran ishlatilishi mumkin .Biroq eng kichik qiymat hamma vaqt xam 0 bo’lmasligini inobatga oling.

6.7. Muammoni yechish: Bosqichma bosqich detallashtirish

Muammoni yechishning eng kuchli strategiyalaridan biri bu bosqichma bosqich detallashtirish jarayonidir. Qiyin vazifani yechish uchun uni oddiyroq vazifalarga bo’ling.So’ngra o’sha oddiyroq vazifani o’zingiz yecha oladigan vazifa bo’lguncha yanada oddiyroq vazifalarga bo’ling.

Mana hayotdagi kunlik muammo jarayonining qo’llanilishi .Ertalab uyg’onasiz va siz ko feolishingiz kerak.Qanday qilib kofe olasiz. Onangiz yoki turmush o’rtog’ingiz olib kela olarmikin deb qaraysiz.Bu xam bo’lmasa, siz o’zingiz kofe tayyorlashingiz kerak.Qanday qilib kofe tayyorlaysiz?

Agar eriydigan kofe bo’lsa,siz eruvchi kofe tayyorlay olasiz.Buni qanday tayyorlaysiz? Oddiygina suvni qaynatib unga eruvchi kofe aralashtirasiz.Suvni qanday qaynatasiz? Agar mikroto’lqinli pech bo’lsa unda siz stakanni suvga to’ldirib mikroto’lqinli pechga 3 daqiqa isitasiz. Yoki choynakni suvga to’ldirib gaz plitasi ustiga qaynaguncha qo’yasiz.Yoki bo’lmasa kofe qaynatishingiz kerak? Qanday qilib? Kofe tayyorlaydigan mashinaga suv quyasiz,filterni ishga tushurasiz, kofeni yanchasiz, uni filterga qo’yasiz va kofe tayyorlaydigan mashinani ishga tushirasiz.Kofeni qanday yanchasiz?Kofe donalarini kofe maydalagich(Kofemolka)ga solib 60 soniyaga tugmani bosasiz.

Chek yozayotganda , chek miqdorini ham raqamda (“$274.15”) va harfli yozuvlarda (ikki yuz yetmish to’rt dollar va o’n besh sent)yozish odatiy hol. Bunday qilish orqali oluvchining pul miqdori oldidan yana bir necha sonlarni

qo’yish kabi aldovlarni oldini oladi. Insonlar uchun bu unchalik qiyin emas,lekin kompyuter 274 ni "ikki yuz yetmish o’rtga"ga aylantiradigan funksiyani qanday tuzadi.

Biz bu funksiyani dasturlashimiz zarur. Mana biz yozmoqchi bo’lgan funksiya ifodasi:

```
/**
Raqamlarni ularning inglizcha nomiga aylantiradi.
@param number manfiy son < 1,000
@return raqam_nomi (e.g., “ikki yuz yetmish to’rt”)
*/
string int_name(int number)
```

Bu funksiya o’z ishini qanday amalga oshiradi?Keling avvaliga oddiy holatga qaraylik.Agar raqam 1 va 9 orasida bo’lsa biz "bir" ..... "to’qqiz"ni hisoblashimiz kerak bo’ladi. Aslida bizga yuzda xam (ikki yuz) yana o’sha xisoblash kerak bo’ladi. Bosqichma bosqich bo’lish jarayonini amalga oshirish orqali biz bu oddiyroq vazifa uchun boshqa funksiya yaratamiz. Funksiyani amalga oshirishdan oldin biz avvalo izoh yozamiz:

```
/**
Raqamni uning inglizcha nomiga aylantiradi.
@param digit 1 va 9 orasidagi butun son
@return the name of digit (“one” ... “nine”)
*/
string digit_name(int digit)
```

if holatidan foydalanib buni amalga oshirish juda oddiy tuyulishi

mumkin. digit_name funksiyasini tugatish uchun boshqa funksiya talab qilinmaydi shu uchun biz amalga oshirish haqida keyinroq o’ylaymiz.

10 va 19 orasidagi raqamlar maxsus holatdir. Ularni "o’n bir ""o’n ikki""o’n uch" va boshqa qatorlarga aylantiradigan aloxida teen_ name funksiyasini olaylik:

/**

10 va19 orasidagi raqamlarni ularning inglizcha nomiga aylantiradi.

@param number 10 va 19 orasidagi butun son

@return raqam nomi (o'n" ... "o'n to'qqiz")

*/

string teen_name(int number)

So'ng raqam 20 va99 orasida deb taxmin qilaylik. Keyin biz o'nliklarni "yigirma", "o'ttiz", …, "to'qson "kabi ko'rsatamiz. Oddiylik va ketma ketlik uchun bu xisoblashni alohida funksiyaga qo'ying:

/**

20 va 99 orasidagi raqamlarning o'nlik qismiga nom beradi.

@param number 20 va 99 orasidagi butun son

@return raqamlarning o'nlik qismining nomi ("yigirma" ... "to'qson")

*/

string tens_name(int number)

Endi raqamni eng kami 20 va eng ko'pi 99 deb taxmin qiling.Agar raqam teng 10 ga bo'linadigan bo'lsa,biz tens_name ni qo'llaymiz va shunday qilamiz.Bo'lmasa o'nliklarni tens_ name bilan, birliklarni digit_name. bilan terib chiqamiz.Agar raqam 100 va 999 orasida bo'lsa, biz sonni ko'rsatamiz, "yuz"so'zi, va qoldig'i oldin ko'rsatilganidek bo'ladi.

Mana bu algoritmning ramziy kodi:

```
part = number (hamon aylantirilishi kerak bo'lgan qism)
name = "" (raqam nomi)
Agar qism >= 100 bo'lsa
name = yuzliklarning qismlardagi nomi + " yuz"
Yuzliklarni qismdan o'chiring.
Agar qism >= 20 bo'lsa
```

```
tens_name(qism)ni nomga qo’shing.
O’nliklarni qismdan o’chiring.
Agar qism >= 10 bo’lsa
tens_name(qism) ni nomga qo’shing.
qism = 0
Agar (qism > 0)
digit_name(qism) ni nomga qo’shing.
```

Bu ramziy kod harfli ifodaga nisbatan ancha qulayliklarga ega. Kattaroq raqamlar uchun solishtirish bilan boshlab u qanday qilib testlarni tartibga solishni ko’rsatadi va qanday qilib kichik raqam keyinchalik qo’shimcha if operatorida qayta ishlanilishini ko’rsatadi.

Boshqa tomondan esa bu ramziy kod qismlarning haqiqiy o’zgarishi haqida aniq bir narsa aytmaydi,faqatgina "yuzliklar nomi"va shunga o’xshashlarni anglatadi.Shuningdek,biz bo’sh joy(probel)haqida aniq bir narsa ayta olmas edik. Hozirgi holatdan ko’rinib turibdiki, kod bo’sh joysiz qatorlarni yaratadi misol uchun, twohundredseventyfour. Asosiy muammoning murakkabligi bilan qiyoslaganda ,bo’sh joy kichik muammodek tuyulishi mumkin. Ramziy kodni kichik detallar bilan tushunarsiz qilmaslik lozim.

Endi ramziy kodni haqiqiy kodga aylantiring.Oxirgi uchta holat oson, chunki ular uchun allaqachon yordamchi funksiyalar yaratilgan:

```
if (part >= 20)
{
name = name + " " + tens_name(part);
part = part % 10;
}
else if (part >= 10)
```

```
{
name = name + " " + teen_name(part);
part = 0;
}
if (part > 0)
{
name = name + " " + digit_name(part);
}
```

Va nixoyat 100 va 999 orasidagi raqamlar holatiga yechim topaylik. Qism < 1000 bo'lgani uchun, qism / 100 birlik son, digit_name ni chaqirish orqali biz uning nomini qo'lga kiritamiz. So'ng biz "hundred" qo'shimchasini qo'shamiz:

```
if (part >= 100)
{
name = digit_name(part / 100) + " hundred";
part = part % 100;
}
```

6.8. O'zgaruvchini aniqlash maydoni va global o'zgaruvchilar

Dasturda bir o'zgaruvchining nomini bir martadan ortiq aniqlash mumkin.O'zgaruvchi nomi ishlatilganda uning qaysi ta'rifga tegishli ekanligini bilishingiz kerak.Ushbu bo'limda biz bir nomning bir necha ta'rifi bilan ishlash qoidalarini muhokama qilamiz.

Funksiya ichida aniqlangan o'zgaruvchi blok oxirigacha aniqlanib, o'sha joydan ko'rinadi. Bu maydon o'zgaruvchini aniqlash maydoni deb ataladi.

Keyingi misolda hajm o'zgaruvchini ko'rib chiqamiz:

```
double kub_hajm(double side_length)
{
```

```
    double hajm = tomon_uzunligi * tomon_uzunligi *
tomon_uzunligi;
    return hajm;
    }
    int main()
    {
    double hajm = kub_hajm(2);
    cout << hajm << endl;
    return 0;
    }
```

Har bir hajm (hajm) o’zgaruvchi aloxida funksiyalarda aniqlangan va ularning aniqlash maydoni bir birini qoplamaydi.

Bitta aniqlash maydonida bitta nomga ega ikkita o’zgaruvchini aniqlash qonuniy emas.Misol uchun quyidagi noqonuniy:

```
    int main()
    {
    double hajm = kub_hajm(2);
    double hajm = kub_hajm(10);
    // XATO: bu maydonda boshqa hajm o’zgaruvchini
aniqlay olmaydi.
    ...
    }
```

Biroq siz kiritilgan blokda bir xil nomli boshqa o’zgaruvchini aniqlay olasiz. Bu yerda biz amount deb nomlangan ikkita o’zgaruvchini aniqlaymiz

```
    double withdraw(double balance, double amount)
```

```
{
if (...)
{
double amount = 10; // amount deb nomlangan bosh
o’zgaruvchi
...
}
...
}
```

Kiritilgan blok ichidan tashqari ,amount parametr o’zgaruvchini aniqlash maydoni to’liq funksiya hisoblanadi. Kiritilgan blok ichida amount o’sha blokda aniqlangan o’zgaruvchiga tegishli bo’ladi.Biz buni ichki o’zgaruvchi tashqi blokda aniqlangan o’zgaruvchini xiralashtiradi,ta’qib qiladi deymiz. Siz funksiya yozayotganingizda,oddiygina bir o’zgaruvchining nomini o’zgartirish orqali bunday chalkash holatlarni oldini olishingiz kerak.

Funksiya ichida aniqlangan o’zgaruvchi mahalliy o’zgaruvchi deb ataladi. C++ da funksiya tashqarisida aniqlangan global o’zgaruvchilarxam bor. Global o’zgaruvchi undan keyin aniqlangan barcha funksiyalarga ko’rinadi. Misol uchun

<iostream> sarlavxasi cin va cout degan global o’zgaruvchilarni aniqlaydi

Mana global o’zgaruvchiga misol:

```
int balance = 10000; // Global o’zgaruvchi
void withdraw(double amount)
{
if (balance >= amount)
{
balance = balance - amount;
```

```
}
}
int main()
{
withdraw(1000);
cout << balance << endl;
return 0;
}
```

balance o’zgaruvchini aniqlash maydoni withdraw va main funksiyalar bo’ylab kengayadi .Umuman olganda global o’zgaruvchilar bu yaxshi g’oya emas. Ko’p marotabali funksiyalar global o’zgaruvchilarni yangilaganda,natijani oldindan aytish qiyin bo’lishi mumkin. Asosan ko’p dasturchilar tomonidan yaratilgan kattaroq dasturlarda har bir funksiyaning aniq va tushunishga oson bo’lishi muxim.Siz o’z dasturingizda global o’zgaruvchilardan qochishingiz kerak.

6.9 Izoh parametri

Agar siz parametr qiymatini o’zgartiradigan funksiya yozmoqchi bo’lsangiz, o’zgartirishga ruxsat berish uchun izohlar parametridan foydalanishingiz kerak .

Avval nima uchun turli xil parametr turi zarurligini tushuntiramiz,keyin esa izohlar parametri uchun sintaksisni ko’rsatamiz.

Bank hisobidan berilgan pul miqdorini qaytib olishni ko’rsatadigan funksiyani ko’ramiz, yetarli pul bo’lganda. Agar pul miqdori yetarli bo’lmasa , $10 jarima solinadi. Funksiya quyidagicha ishlatiladi:

```
    double pul_miqdori = 1000;
withdraw(pul_miqdori, 100); // Xozir pul_miqdori 900
    withdraw(pul_miqdori, 1000); // Pul yetishmaydi.
Endi pul_miqdori 890
```

Mana birinchi harakat:

```
void withdraw(double balance, double amount) //
Ishlamaydi
{
const double PENALTY = 10;
if (balance >= amount)
{
balance = balance - amount;
}
else
{
balance = balance - PENALTY;
}
}
```

Biroq bu ishlamaydi. Keling withdraw(pul_miqdori, 100)—chaqiruv funksiyasini qarab chiqaylik.

Funksiya boshlanganda, parametr o’zgaruvchi balance hosil bo’ladi 1 pul_miqdoriga o’xshab o’sha qiymatda o’rnatiladi va amount 100 deb belgilanadi. Keyin balance o’zgartiriladi.Albatta bu o’zgartirishning pul_miqdoriga ta’siri yo’q, balance bu aloxida o’zgaruvchi.Funksiya qaytganda balance unitiladi,va pul_miqdori dan xech qanday pul qaytarib olinmaydi.

Parametr o’zgaruvchi "balance " parametr qiymati deb ataladi,chunki u berilgan parametr qiymati bilan boshlang’ich holatga keltirilgan.Yozgan barcha funksiyalarimiz parametr qiymatini ishlatadi.Bu holatda biz balance ning pul_miqdoriga o’xshab qiymatga ega bo’lishini xoxlamaymiz.Biz balancening haqiqiy o’zgaruvchi harrys_ account ga tegishli bo’lishini xoxlaymiz. (yoki

joes_account yoki chaqiruvda berilgan xoxlagan o’zgaruvchi). O’sha o’zgaruvchining tarkibi yangilanishi kerak.

Funksiya chaqiruvida berilgan o’zgaruvchini yangilash uchun izohlar parametridan foydalanasiz.Biz izohlar parametriga balance tuzganimizda, balance bu yangi o’zgaruvchi emas balki mavjud bo’lgan o’zgaruvchiga izoxdir.

bal-ance dagi har bir o’zgarish bu o’zgaruvchidagi o’zgarishdir ,qaysikim balance o’sha aniq chaqiruvga tegishli bo’ladi.

Izoh parametrini aniqlash uchun siz & ni joylashtirasiz va so’ng nomni terasiz.

```
void withdraw(double& balance, double amount)
```

Yozilgan double& “double ga izoh” deb o’qiladi,yoki yana qisqaroq , “double ref”. withdraw funksiyasining endi ikkita parametr o’zgaruvchisi mavjud: birinchisi “double ref” va boshqa qiymat parametri double.Funksiya tanasi o’zgarmaydi.

```
balance = balance - amount;
```

endi funksiyaga uzatilgan o’zgaruvchini o’zgartiradi.

Mavzuga doir test savollar:

1. Quyidagi ifoda qanday natija chiqaradi?
   func() { int a = 10; }
   main(){
   int x2 = a + 1;
   cout<< x2;
   }
   a) 11
   b) 1
   c) 12
   d) * kompilyasiyada xatolik
2. Quyidagi ifoda qanday natija chiqaradi?
   int func() { int a = 10; }
   main(){
   int a=2, b=3;
   int c=func();
   cout<< c; }
   a) 11

b) 1
c) 12
d) * kompilyasiyada xatolik

3. Quyidagi ifoda qanday natija chiqaradi?

```
int func(int a, int b) { return a+b; }
main(){
int a=2, b=3;
int c=func(a,b);
cout<< c;
 }
```

a) 11
b) 5
c) 12
d) kompilyasiyada xatolik

4. Quyidagi ifoda qanday natija chiqaradi?

```
main(){
int a=2, b=3;
int c=func(a,b);
cout<< c;
 }
int func(int a, int b) { return a+b; }
```

a) 11
b) 5
c) 12
d) kompilyasiyada xatolik

5. Quyidagi ifoda qanday natija chiqaradi?

```
void func(int a, int b) { return a+b; }
 main(){
 int a=2, b=3;
 int c=func(a,b);
 cout<< c;
  }
```

a) 11
b) 5
c) 12
d) kompilyasiyada xatolik

Nazorat savollari:

1. Funsiyaning nima?

2. Funksiyaning vazifasi nima?
3. Nechta turdagi funksiyalar mavjud?
4. Funsiyaning turi bormi?
5. Qandey funksiyalar qiymat qaytarmaydi?
6. Qandey funksiyalarni matematik formulalarda ishtib bo'lmaydi?
7. Qandey funksiyalarda return kalit so'zini ishlatib bo'lmaydi?
8. Void turidagi funksiyalarning maqsadi nimadan iborat?
9. Funksiya prototipi nima va uning maqsadi?
10. Rekursiv funksiya nima va uning maqsadi?
11. Funksiyadan qandey masalalarni yechishda foydalaniladi?
12. Funksiya turi nima va uning natijasichi?
13. Funksiya parametric nima vazifani bajaradi?
14. Qandey o'zgaruvchilar paramet bo'la olmaydi?
15. Parametrlaning shartlari qandey?
16. Qandey parametrlar qiymat obkeladi?
17. Qandey parametrlar adres obkeldi?
18. Parametrsiz funksiyalar nima uchun kerak?
19. Sikl jarayonlarini tashkil etish uchun qandey funksiyalarni tashkil etish kerak?
20. Rekursiv funksiyaning asosisy shartlarini bilasizmi?

7. Bir o’lchovli massivlar

Ko’p dasturlarda, siz katta qiymatli sonlarni yig’ishingiz kerak bo’ladi. Bu maqsadda siz massiv va vektorlardan standart C++ da foydalanishingiz mumkin.Massivlar C++ dastur tilining asosiy tuzilmasidir. C++ standart kutubxonasi o’lchami qayd etilmaydigan kolleksiya bilan ishlash mobaynida qulay alternativa sifatida vektorli konstruksiyani taklif etadi.Ushbu bobda siz massivlar, vektorlar va ularni qayta ishlash uchun umumiy algoritmlar bilan tanishasiz.

7.1 Massivlar

Biz bu bobni massiv ma’lumotlar turini tanishtirishdan boshlaymiz. Massivlar bir necha qiymatlarni yig’ish uchun C++ da asosiy mexanizm hisoblanadi. Quyidagi bo’limlarda siz massivlarni qanday aniqlashni va massiv elementlaridan qanday foydalanishni o’rganib olasiz.

7.1.1 Vektorlarni aniqlash

Deylik siz qiymatlar ketma -ketligini o’quvchi va ketma ketlikni chop etuvchi dastur yozmoqchisiz, buning uchun siz eng katta qiymatni quyida berilgan ko’rinishda belgilang:

32
54
67.5
29
34.5
80
115 <= eng katta qiymat
44.5
100
65

Bu qiymatlarni barchasini ko’rmasdan turib, siz qaysi birini eng katta qiymat

deb belgilash kerakligini bilmaysiz.Oxir oqibat, oxirgi qiymat eng kattasi bo'lishi mumkin. Shuning uchun, dastur chop etishdan oldin birinchi navbatda barcha qiymatlarni saqlab olishi kerak. Har qaysi qiymatni alohida o'zgaruvchida oddiygina saqlasangiz bo'lmasmidi? Bilsangiz, o'nta o'zgaruvchini ya'ni qiymat 1 (value1), qiymat 2 (value2), qiymat 3 (value3), …, qiymat 10(value10)larni o'nta o'zgaruvchida saqlovchi o'nta kirituvchi mavjud. Biroq, bunday o'zgaruvchilarning ketma - ketligi foydalanish uchun noqulay. Siz, har qaysi o'zgaruvchi uchun oddiy kodni o'n marotaba yozishishga to'g'ri keladi. Bu muammoni hal etish uchun massivdan foydalaning: qiymatlar ketma – ketligini saqlovchi struktura (tuzilma).

Qo'sh qiymatlar [10];

Qo'sh massiv turiga kiruvchi o'zgaruvchan qiymatlarga berilgan ta'rif. Ya'ni, qiymatlar suzuvchi nuqtadagi sonlar ketma - ketligini saqlaydi. [10] massiv miqdorini belgilaydi. (1- shaklga qarang.) Kompillyasiya jarayonida, ma'lum bo'lishicha, massiv miqdori o'zgarmas bo'lishi shart.

Massivni aniqlayotganda ilk qiymatlarni tasniflashingiz mumkin. Masalan:

qo'sh qiymatlar [] = { 32, 54, 67.5, 29, 34.5, 80, 115, 44.5, 100, 65 };

ilk qiymatlarni taqdim etayotkaningizda, massiv o'lchamlarini tasniflashingiz shart emas. Kompilyator qiymatlarni hisoblash jarayonida miqdorni aniqlab oladi.

1-ЖАДВАЛ Массивларни аниқлаш	
`int numbers[10];`	ўнта бутун сондан иборат массив.
`const int SIZE = 10;` `int numbers[SIZE];`	миқдор учун номланган ўзгармасдан
⚠ `int size = 10;` `int numbers[size];`	Диққат: Стандарт C++да миқдор ўзгармас бўлиши шарт.Массивнинг бу таърифи барча компиляторлар билан ишламайди.
`int squares[5] = { 0, 1, 4, 9, 16 };`	
`int squares[] = { 0, 1, 4, 9, 16 };` `int squares[5] = { 0, 1, 4 };`	Агар бошланғич қийматни тақдим этсангиз, массив миқдорини тушириб қолдиришингиз мумкин.Миқдор бошланғич соннинг қийматига кўра ўрнатилади.
`string names[3];`	3 элементли string туридаги массив

7.1.2 Massiv elementlarini kiritish

Massivdv saqlanovchi qiymatlar uning elementlari deb ataladi. Har qaysi element indeks deb ataluvchi o’rin soniga ega.Qiymatlar massiviga qiymat kiritish uchun siz qaysi indeksdan foydalanishni xoxlashingizni aniqlashingiz lozim. Bu [] operatori tomonidan bajariladi: qiymatlar [4] = 34.5; indeksi 4 ga teng element 34.5 bilan to’ldirilgan. Quyidagi buyruq yordamida indeksi 4 element tarkibini ko’rsatishingiz mumkin:

```
cout << qiymatlar[4] << endl;
```

ko’rib turganingizdek, [4] element qiymatlari qo’sh turdagi har qanday o’zgaruvchan kabi qo’llanilishi mumkin.

C++ da, massiv joylashuvining hisobi sizni lol qoldiradigan tarzda hisoblanadi. Agar 2- shaklga sinchiklab e’tibor qaratsangiz, siz beshinchi element [4] qiymatlarni o’zgartirganda to’ldirilishini ko’rishingiz mumkin. C++ da, massivlar elementlari 0 dan boshlab raqamlanadi. Ya’ni, massiv qiymatlari uchun qonuniy elementlari:

`qiymatlar[0],` birinchi element

qiymatlar[1], ikkinchi element qiymatlar[2], uchinchi

element qiymatlar[3], to’rtinchi element

qiymatlar[4], beshinchi element

...

qiymatlar[9], o’ninchi element

Siz 7 Bobda nimaga C++ da ushbu raqamlash sxemasi tanlanganligi bilan tanishasiz.Qiymatlar indeksiga nisbatan ehtiyot bo’lishigiz kerak.

Massivda mavjud bo’lmagan elementga kirishga urinish jiddiy xato qisoblanadi.

Masalan: agar qiymat 20 ta elementdan iborat bo’lsa, sizga [20] qiymatlarni kiritishga ruxast berilmaydi.Mavjud indeks ko’lamida bo’lmagan elementga kirishga urinish tasodifiy xato deb ataladi.Kompilyator bu turdagi xatoni topmaydi. Xatto ishlayotgan dastur xato yo’q deb xabar bersa ham. Agar siz shunday xatolik qilgan bo’lsangiz, siz jimgina boshqa xotirada o’qing yoki qayta yozing.

Natijada, dasturingizda tasodifiy xatoliklar bo’lishi mumkin va bu deyarli eng ko’p tarqalagan xatolar quyidagilardir:

qo’sh qiymatlar [10];

```
cout << qiymatlar[10];
```

o’nta elementdan iborat massivda [10] qiymati mavjud emas.Qonuniy elementlar ko’lami 0 dan 9 gacha.

Massivning barcha elementlariga tashrif buyurish uchun indeks uchun o’zgaruvchidan foydalaning. Aytaylik. qiymat o’nta elementga ega va i butun o’zgaruvchan 0, 1, 2, va boshqalar 9 gacha bo’lgan qiymatni oldi. Keyin [i] ifoda qiymati har qaysi qiymatni navbat bilan chiqaradi. Misol uchun: bu takrorlash barcha elementlarni ko’rsatadi.

```
for (int i = 0; i < 10; i++)
{
```

```
cout << qiymatlar[i] << endl;
}
```

E’tibor bering, takrorlash holatida indeks 10dan kichik, chunki [10] qiymatiga muvofiq keladigan element yo’q.

7.1.3 Qisman to’ldirilgan massiv

Massiv ishlash jarayonida miqdorini o’zgartirishi mumkin emas va bu oldindan qancha element kerakligini bilmaslikdagi muammodir.Bunday holatda, saqlashingiz kerak bo’lgan edementlarning maksimal sonini yaxshigina tasavvur qilishingiz kerak. Biz bu miqdorini hajm deb ataymiz. Misol uchun. biz ba’zida o’nta miqdordan ko’p lekin 100dan ko’p bo’dlmagan qiymatlarni saqlashga qaror qilamishimiz mumkin:

```
doimiy boshl.HAJM = 100;
qo’sh qiymatlar [Xajm];
```

oddiy dastur ishlaganda, massivning faqatgina bir qismi haqiqiy elementlar bilan band bo’ladi.Bunday massiv qisman to’ldirilgan massiv deb nomlanadi. Aslida qancha element ishlatilayotganini hisoblovchi hamroh o’zgaruvchini saqlashingiz kerak. 3- shakldagi hamroh o’zgaruvchini biz joriy miqdor deb ataymiz.

Quyidagi takrorlash qiymatlarni yig’adi va massiv qiymatlarini to’ldiradi.

```
int current_size = 0;
double input;
while (cin >> input)
{
if (current_size < CAPACITY)
{
qiymatllar[current_size] = input;
```

```
current_size++;
}
}
```

Ushbu takrorlash oxirida, joriy miqdor massivdagi aniq elementlar sonini o’zida saqlaydi. Agar massiv miqdori hajmga tenglashsa, siz kirishni to’xtashingizga to’g’ri keladi. To’plangan massiv elementlarini qayta ishlash uchun siz hajmni emas, balki yana hamroh o’zgaruvchini qo’llaysiz. Bu takrorlash qisman to’ldirilgan massivni chop etadi:

```
for (int i = 0; i < current_size; i++)
{
cout << qiymatlar[i] << endl;
}
```

Mustaqil nazorat:

1. Birinchi besh sodda raqamlarni o’z ichiga olgan butun son massivini aniqlang.
2. Deylik, massivdagi sodda raqamlar mustaqil nazorat 1 da tasvirlangandek belgilangan deb. Quyidagi takrorlashni bajargandan so’ng uning tarkibi qanday bo’ladi ?

```
for (int i = 0; i < 2; i++)
{
primes[4 - i] = primes[i];
}
```

3. Deylik, massivdagi sodda raqamlar mustaqil nazorat 1 da tasvirlangandek belgilangan deb. Quyidagi takrorlashni bajargandan so’ng uning tarkibi qanday bo’ladi ?

```
for (int i = 0; i < 5; i++)
```

```
{
primes[i]++;
}
```

4. Ta’rif berilgan

```
const int CAPACITY = 10;
double values[CAPACITY];
```

nol sonini eng quyi va eng yuqori haqiqiy indeksli massiv qiymatlarning elementlariga kiritish uchun dastur gapini yozing.

5. Mustaqil nazorat 4 da aniqlangan massiv berilgan, oxigi elemendan boshlab, massiv qiymatlarining elementlarini teskari tartibda chop etish uchun takrorlash yozing.
6. O’n qiymatli satr turidan iborat so’zlar deb nomlanuvchi massivni aniqlang.
7. **HA va YO’Q ikki satrdan iborat massivni aniqlang.**

Tasodifiy fakt. Ilk internet virusi

1988 yilning Noyabr oyida Kornel Universiteti talabasi Robert Morris AQSh bo’ylab internetga ulangan 6000 ta kompyuterlarni zaralagan shu nomdagi virusni tarqatdi. Natijada, o’n minglab kompyuter foydalanuvchilari elektron xabarlarini o’qiy olmadilar va xattoki kompyuterdan foydalana ham olmadilar. Barcha asosiy Universitetlar va ko’plab katta korxona va tashkilotlar ham bundan zaralandilar. (Internetning ko’lami u davrda kichkina edi.) Bu hujumni uyushtirgan virus

"Worm" virusi deb nomlandi.Ushbu virus dasturi internet orqali bir kompyuterdan ikkinchisiga o’rmalagan tarzda yuqadi. Ushbu virus UnIX operasion tizimidagi dasturga ulanib, tarmoqda ma’lum kompyuterda hisob raqamiga ega bo’lgan foydalanuvchi haqida ma’lumot olishga harakat qiladi. Boshqa dasturlar kabi UnIXdagi dastur C++ dasturlash tilida yozilgan bo’lib, bu tilda massivlar belgilangan miqdorga ega. Qidirilayotgan

foydalanuvchining nomini (deylik, wal-ters@cs.sjsu.edu) saqlosh uchun

finger dasturi, hyech kim bunday uzun kiritishni huch qachon qo'llamaydi degan tahminda, 512 ta belgidan iborat massivni ajratgan. Afsuski, C, C++ga dasturchi dangasa bo'lgan va kiritish belgilarini o'zida saqlagan massiv kiritishni ushlab turish uchun yetarlicha kattami yo'qmi tekshirmagan. Shu sababali worm dasturi maqsadli tarzda 512 belgili dasturni 536 bit bilan to'ldirgan. Ortiqcha 24 bayt esa hujumchiga ma'lum bufer satridan so'ng, saqlangan qaytish manzilini qayti yozadi. Ushbu funksiya niqoyasiga yetkanda esa, u o'zini chaqiruvchi dasturiga qaytmasdan balki Worm dasturi taklif etgan kodga qaytgan. bu kod finger dasturidek super foydalanuvchi imtiyozlari ostida ishlagan va worm dasturiga masofadagi tizimiga kirishga imkon beradi. Ushbu finger dasturini yozgan dasturchi vijdonli bo'lganda edi, bunday hujum yuz bermagan bo'lar edi. Ko'rinib turibdiki, buzibkirish bu muallif tomonidan uyutirilgan lekin kompyuterlarni ishdan chiqargan narsa bu uzluksiz qayta zaralanish oqibatidagi texnik nuqson bo'lgan. Morris 3 yil shartli ozodlikdan mahkum etilgan, u 400 soat jamoat ishlarini bajarilishi va $10,000 miqdordagi jarima to'lashi kerak edi.oxirgi moliyaviy ma'lumotlarni o'g'irlashadi yoki hujumga uchragan kompyuterdan elektron pochtaga spam jo'natishadi. Afsuski, buferda nosozliklar ko'payib ketganligi va sifatsiz yozilgan dasturlar sababli, unday hujumlar yuz berishi davom etadi.

7.2 Umumiy massiv algoritmlari

Quyidagi bo'limlarda biz, qiymatlar ketma- ketligini qayta ishlash uchun ba'zi eng keng tarqalgan algoritmlarni muhokama qilamiz.Biz algoritmlarni shunday taqdim etamizki, siz ularni to'liq va qisman to'ldirilagan massivlar hamda (6.7 bo'limda tanishtiradigan) vektorlar yordamida foydalanishingiz mumkin. Qiymatlar miqdori (size of values) ifodasini ishlatganimizda, siz uni massivda elementlar miqdorini anglatuvchi doimiy yoki o'zgaruvchan bilan almashtirishingiz kerak. (yoki values.size() if values is a vector ifodasi.)

```
for (int i = 0; i < sizeof(qiymatlar); i++)
{
```

```
qiymatlar [i] = 0;
}
```

So’ngra, keling kvadratlar massivini 0, 1, 4, 9, 16, va h.z sonlar bilan to’ldiramiz. E’tibor bering, 0 indeksli element 02 ni , 1 indeksli element 12 va h.zni o’z ichiga oladi.

```
for (int i = 0; i < sizeof(kvadrat); i++)
{
kvadrat[i] = i * i;
}
```

7.2.1 Nusxa ko’chirish

Ikkita massivni ko’rib chiqamiz:

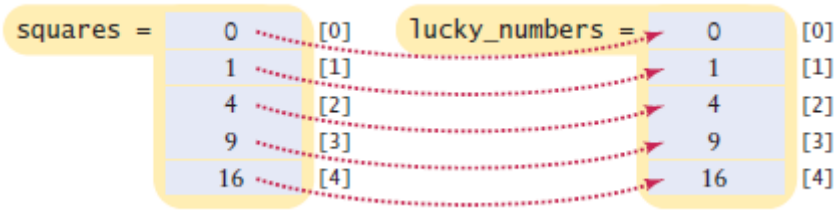
```
int kvadrat[5] = { 0, 1, 4, 9, 16 };
int omadli_raqamlar[5];
```

Hozir aytaylik, siz birinchi massivdagi barcha qiymatlarni ikkinchisiga ko’chirmoqchisiz. Quyidagi ifoda xato hisoblanadi:

```
omadli_raqamlar = kvadrat; // Error
```

C++da siz bir massivni ikkinchisiga o’tkaza olmaysiz. Uning o’rniga siz barcha elementlarni ko’chirish uchun takrorlashdan foydalanishingiz kerak:

```
for (int i = 0; i < 5; i++)
{
omadli_raqamlar [i] = kvadrat[i];
}
```



7.2.2 Yig'indi va o'rtacha qiymat

elementlarining yig'indisini hisoblash kodi berilgan.

```
double jami = 0;

for (int i = 0; i < sizeof(qiymatlar); i++)

{

jami = jami + qiymatlar[i];

}
```

O'rtacha qiymatni topish uchun elementlar miqdorini bo'ling:

```
double o'rtacha = jami / sizeof(qiymatlar);
```

miqdor nol emasligini tekshirishni unutmang.

7.2.3 Eng katta va eng kichik

O'zida eng katta element uchun o'zgaruvchini saqlovchi va massivlarni joriy etishda uchratgan, 4.7.4 - bo'limdagi algoritmni qo'llang:

```
double eng_kattasi = values[0];

for (int i = 1; i < sizeof(qiymatlar); i++)

{

if (qiymatlar[i] > eng_kattasi)

{

eng_kattasi = qiymatlar[i];

}
```

```
}
```

Biz eng katta qiymatni [0] qiymati bilan inisializasiyalashtirganligimiz tufayli, takrorlash 1 dan boshlanishiga e’tibor qarating. Eng kichik qiymatni hisoblash uchun solishtirishni orqaga qaytaring. Bu algoritmlar massiv kamida bitta elementni o’z ichiga olishni talab qiladi.

7.2.4 Element ajratuvchilari

Siz element to’plamlarini namoyish etganingizda, siz odatda ularni vergul yoki quyidagicha vertikal chiziqlar bilan bo’lishni hohlaysiz:

```
1 | 4 | 9 | 16 | 25
```

E’tibor bering, ajratuvchi sonlarga qaraganda bittaga kam, dastlabki(0 indeksli) dan tashqari har qaysi elementdan oldin ajratuvchini yozing.

```
for (int i = 0; i < sizeof(qiymatlar); i++)
{
if (i > 0)
{
cout << " | ";
}
cout << qiymatlar[i];
}
```

7.2.5 Bir chiziqli qidiruv

Uni o’rnini almashtirish yoki olib tashlash uchun siz tez- tez element o’rnini topishingiz kerak bo’ladi. Massivning oxiriga kelmaguncha yoki o’xshashlik topmaguningizcha, barcha elementlarni ko’rib chiqing. Quyida biz 100ga teng bo’lgan birinchi element o’rnini qidiramiz.

```
bool found = false;
while (pos < sizeof(qiymatlar) && !found)
```

```
{
if (qiymatlar [pos] == 100)
{
found = true;
}
else
{
pos++;
}
}
```

Agar topilma to’g’ri bo’lsa, joy birinchi moslikning o’rni hisoblanadi.

7.2.6 Elementni olib tashlash

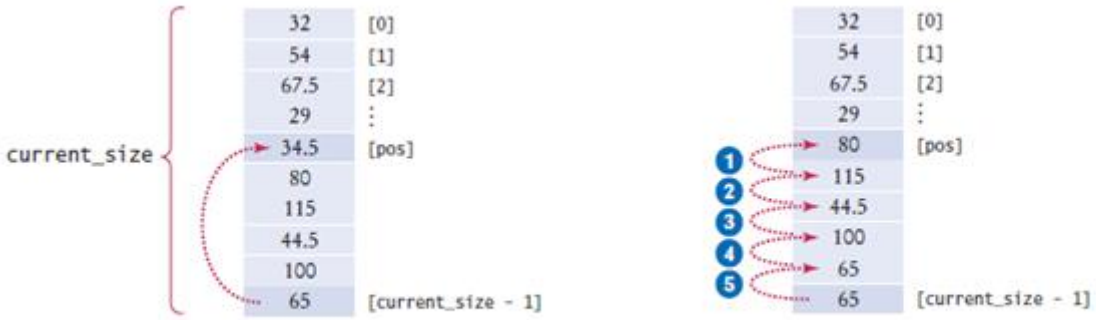
Joriy miqdori o’zgaruvchan joriy miqdorda saqlanovchi qisman to’ldirilgan mas siv miqdorini ko’rib chiqamiz. Deylik, siz pos indeksli elementni qiymatdan olib tashlamoqchisiz.Agar elementlar hyech qanday aniq tartibda bo’lmasa, bu vazifani amalag oshirish yengil kechadi.Shunchaki, oxirgi elementdan olib tashlangan elementni qaytadan yozing.So’ng, miqdorni kuzatuvchi o’zgaruvchini kamaytiring.

```
qiymatlar[pos] = qiymatlar[current_size - 1];
current_size--;
```

Agar elementlar tartibi muhim bo’lsa vaziyat yanada murakkablashadi. Bunday holda siz barcha elementlarni keyingi indeksi pastroq elementga almashtirishingiz kerak. So’ng massivni miqdorini saqlagan holda o’zgaruvchini kamaytirishingiz kerak.

```
for (int i = pos + 1; i < current_size; i++)
{
```

```
values[i - 1] = values[i];
}
current_size--;
```



7.2.7 Element kiritish

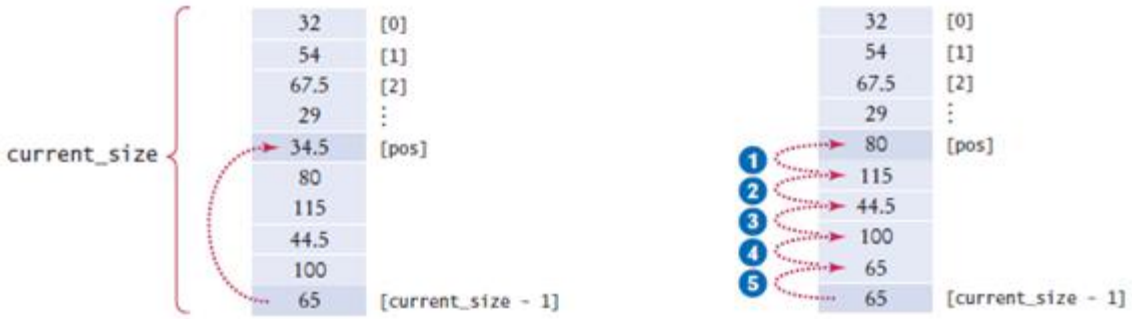
Agar element tartibi muhim bo'lmasa, siz oddiygina miqdorni kuzatuvchi o'zgaruvchini ko'paytirib yangi elementni oxiriga kiritishingiz mumkin .If the order of the elements does not matter, you can simply insert new elements at the end, incrementing the variable tracking the size. (8 - shaklaga qarang.) Qisman to'ldirilgan massiv uchun:

```
if (current_size < CAPACITY)
{
current_size++;
qiymatlar[current_size - 1] = new_element;
}
```

Ketma - ketlikning o'rtasida muayan holatda element kiritish ko'proq ishni talab qiladi. Birinchidan, joriy miqdorni o'zida ushlab turgan o'zgaruvchini ko'paytiring. Keyin, barcha elementlarni qo'shish o'rnidan yuqoriroq indeksga ko'chiring. Nihoyat, yangi elementni joylashtiring.Quyida qisman to'ldirilgan massiv uchun kod berilgan:

```
if (current_size < CAPACITY)
{
current_size++;
for (int i = current_size - 1; i > pos; i--)
{
qiymatlar[i] = qiymatlar[i - 1];
}
qiymatlar[pos] = new_element;
}
```

Harakatlarning tartibiga e’tibor qarating: Elementni olib tashlaganingizda, birinchi bo’lib keyingi elementni pastroq indeksga o’zgartiring. Keyin esa shu ketma - ketlikda massivni oxiriga yetib borguningizgacha davom ettiring. Yangi element kiritishda siz massivni oxiridan boshlaysiz. Shu elementni yuqoriroq indeksga o’zgartiring va keyin undan oldingisiga va xakozo nihoyat qo’shish joyiga kelguningizgacha.



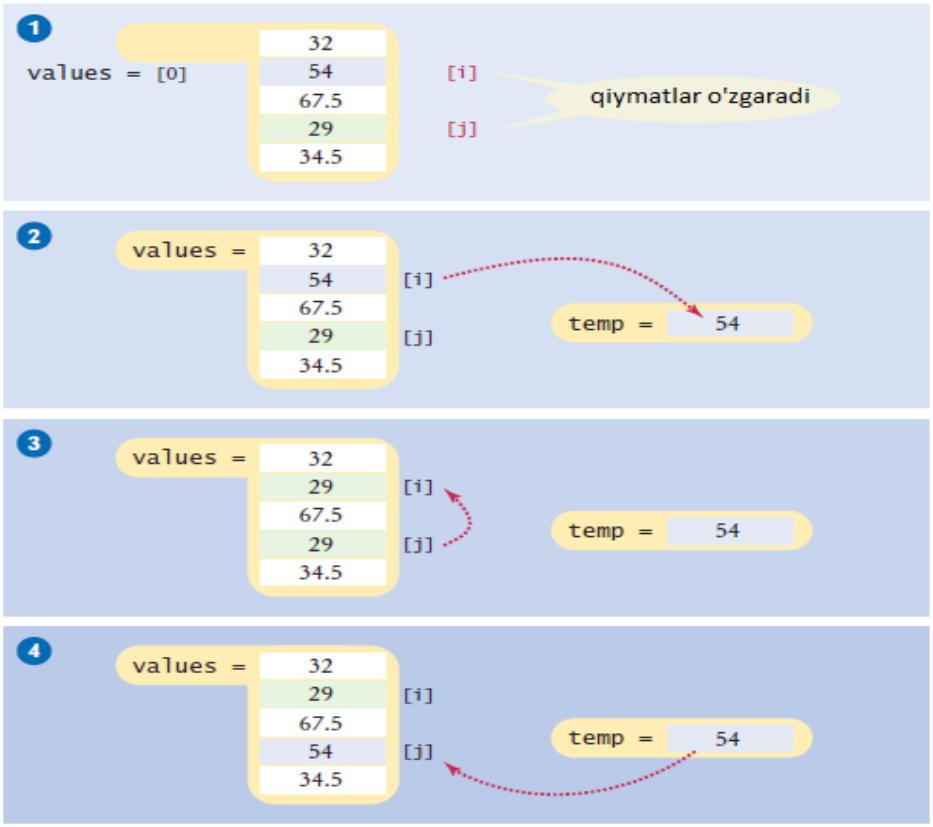
Mashtirish vazifasini ko’rib chiqamiz. Biz [i] qiymatlarini [j] qiymatlariga o’rnatishni hoxlar edik, lekin bu ayni paytda [i]qiymatlarida saqlangan qiymatni qaytadan yozganligi sababli biz birinchisini saqlashni istaymiz.

```
double temp = qiymatlar[i];
qiymatlar[i] = qiymatlar[j];
```

Endi biz [j] qiymatlarini saqlangan qiymatga o’rniatishimiz mumkin.

```
qiymatlar[j] = temp;
```

quyida buni ko’rishingiz mumkin.



7.2.8 Kiritishni o’qish

```
double qiymatlar[NUMBER_OF_INPUTS];

for (i = 0; i < NUMBER_OF_INPUTS; i++)

{

cin >> qiymatlar[i];
```

Ikki elementni olib tashlash uchun, sizga doimiy o’zgaruvchi kerak bo’ladi. Ikkita elementni almashtirishda vaqtinchalik o’zgaruvchidan foydalaning. Biroq tasodifiy kiritish sonlarini o’qish kerak bo’lganda bu usul ish bermaydi.

Bunday holatda, massivga qiymatlarni kiritish oxiriga yetmaguncha qo’shib boravering.

```
double qiymatlar [CAPACITY];
int current_size = 0;
double input;
while (cin >> input)
{
if (current_size < CAPACITY)
{
qiymatlar[current_size] = input;
current_size++;
}
}
```

Endi qiymat qisman to'ldirilgan massiv bo'ldi va joriy miqdorning o'zgaruvchi hamrohi kiritish qiymatlar soniga o'rnatiladi.Bu takrorlash massivga mos kelmagan har qanday kiritish qiymatlarini chiqarib tashlaydi.Hajmga yetkanda qiymatlarni yangi kattaroq massivga ko'chirish yaxshiroq yondoshuv bo'ladi. Quyidagi dastur shu bobning avvalida oldimizga qo'ygan vazifa ya'ni, kiritish ketma - ketligida eng katta qiymatni belgilashni hal etadi.

```
#include<iostream>
using namespace std;
int main(){
    const int CAPACITY=1000;
    double qiymatlar[CAPACITY];
    int current_size=0;
    cout<<"Qiymatlarni kiriting, Q chiqish
uchun"<<endl;
```

```
double input;
while(cin>>input){
    if(current_size< CAPACITY)
    {
        qiymatlar[current_size]=input;
        current_size++;
    }
}
    double eng_kattasi=qiymatlar[0];
    for(int i=1; i<current_size;i++)
    {
      if(qiymatlar[i]>eng_kattasi)
      {
        eng_kattasi=qiymatlar[i];
      }
    }
    for(int i=1; i<current_size;i++)
    {
        cout<<qiymatlar[i];
        if(qiymatlar[i]== eng_kattasi)
        {
            cout<<" Eng katta qiymati";
        }
        cout<<endl;
```

```
            }
        }
```

7.3 Massiv va funksiyalar

Bu bo'limda massivlarni qayta ishlovchi funksiyalarni qanday yozishni o'rganamiz. Massivda qiymatlarni qayta ishlovchi funksiya, massivdagi haqiqiy (aniq) valid elementlarning sonini bilishi kerak.Misol uchun, bu yerda massivdvgi barcha elementlarning yig'indisini hisoblvchi yig'indi funksiyasi berilgan: shqo'd gsrgt:

```
double sum(double qiymatlar[], int size)
{
double total = 0;
for (int i = 0; i < size; i++)
{
total = total + qiymatlar[i];
}
return total;
}
```

O'zgaruvchan parametrli massiv uchun maxsus sintaksisga e'tibor qarating. O'zgaruvchan parametrli massivni yozayotganda siz bo'sh []massivni parametr nomidan so'ng qo'yasiz. Massivning miqdorini qovus ichida ko'rsatmang. Funksiyani chaqirishda massivning miqdori hamda nomini berib o'ting. Misol uchun,

```
double NUMBER_OF_SCORES = 10;
double scores[NUMBER_OF_SCORES]
= { 32, 54, 67.5, 29, 34.5, 80, 115, 44.5, 100, 65
};
```

```
double total_score = sum(scores, NUMBER_OF_SCORES);
```

Siz funksiyaga kichikroq miqdorni ham uzatishingiz mumkin.

```
double partial_score = sum(scores, 5);
```

Bu chaqiruv ballar massivining dastlabki besh element yig'indisini hisoblaydi. Shuni unutmanki, funksiyani massivda qancha element borligini bilishning hyech qanday ilojisi yo'q. U shunchaki chaqiruvchi yetkazib beradigan miqdorga tayanadi.

Massiv parametrlari har doim tayanch parametrlaridir (Sababini 7 - bobda bilib olasiz). Funksiyalar massiv parametrlarini o'zgartirishi mumkin va o'sha o'zgarishlar funksiyaga yuborilgan massivga ta'sir ko'rsatadi. Misol uchun, quyidagi ko'paytirish funksiyasi massivdagi hamma elementlarni yangilaydi.

```
void kopaytirish(double qiymatlar[], int size,
double factor)
{
for (int i = 0; i < size; i++)
{
qiymatlar[i] = qiymatlar[i] * factor;
}
}
```

Bu holatda siz tayanch parametrlarini ifodalash uchun & belgidan foydalanmaysiz.Massiv funksiya parametrlari bo'lsada, ular qaytishning funksional turlari bo'la olmaydi.Agar funksiya bir necha qiymatlarini hisoblasa,funksiya chaqiruvchisi natijani saqlash uchun o'zgaruvchan massiv parametrini taqdim etishi shart.

```
void kvadrat(int n, int result[])
{
```

```
for (int i = 0; i < n; i++)
{
result[i] = i * i;
}
}
```

Funksiya massivning miqdorini o’zgartirganda, u chaqiruvchiga chaqiruvdan so’ng massivda qancha element borligini ko’rsatishi kerak. Buning eng oson usuli yangi miqdrni qaytarishdir. Misol uchun quyida, massivga kiritish qiymatlarini qo’shuvchi funksiya berilgan.

```
int read_inputs(double inputs[], int capacity)
{
int current_size = 0;
double input;
while (cin >> input)
{
if (current_size < capacity)
{
inputs[current_size] = input;
current_size++;
}
}
return current_size;
```

E’tibor beringki, bu funksiya massivning hajmini ham bilishi kerak.Umuman olganda massivga element qo’shuvchi funksiya uning hajmini bilishi kerak. Siz bu funksiyani quyidagicha chaqirishingiz mumkin:

```
const int MAXIMUM_NUMBER_OF_VALUES = 1000;
double qiymatlar[MAXIMUM_NUMBER_OF_VALUES];
int current_size = read_inputs(qiymatlar,
MAXIMUM_NUMBER_OF_VALUES);
```

Shu bilan bir qatorda siz miqdorni tayanch parametri sifatida o'tkazishingiz mumkin. Ushbu usul mavjud massivni o'zgartiruvchi funksiyalar uchun ancha mosdir.

```
void append_inputs(double inputs[], int capacity,
int& current_size)
{
double kiritish;
while (cin >> kiritish)
{
if (current_size < capacity)
{
inputs[current_size] = kiritish;
current_size++;
}}
}
#include<iostream>
using namespace std;
int mas_uqish_va_yozish(double mas[], int
uzunligi){
    int element =0;
        cout<<"Qiymatlarni kiriting, Q chiqish
```

```
uchun"<<endl;
        bool more=true;
        while(more){
         double kiritish;
         cin>>kiritish;
         if(cin.fail()){
            more=false;
         }
         else if(element< uzunligi)
         {
           mas[element]=kiritish;
           element++;
         }
        }
      return element;
    }
    void kopaytirish(double mas[], int elementlar_soni,
double factor){
        for(int i=0; i<elementlar_soni; i++)
        {
            mas[i]=mas[i] * factor;
        }
    }
    void mas_chiqarish(double mas[], int
elementlar_soni){
```

```
        for(int i=0; i<elementlar_soni; i++)
        {
            cout<<mas[i]<<", ";
        }
    }
    int main(){
        const int uzunligi=1000;
        double mas[uzunligi];
        int elementlar_soni=mas_uqish_va_yozish(mas,
uzunligi);
        kopaytirish(mas, elementlar_soni, 2);
        mas_chiqarish(mas, elementlar_soni);
    }
```

Mavzuga doir test savollari:

1. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
 int mas[]={1,2,3,4,5,6,7,8,9};
 cout<<mas[2]<<endl;
}
```

a) 1
b) 2
c) 3
d) 4

2. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
 int mas[100] ={0}; int a=2;
mas[a+2]=2;
 cout<<mas[a+2-1]<<endl;
 }
```

a) 1
b) 2
c) 3

d) 0

3. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){ const int size=10;
 int mas[]={1,2,3,4,5,6};
  for(int i=0; i<4;i++){
 cout<<mas[i]<<" ";
 }
```

a) 1 2 3 4
b) 2 3 4 5
c) 1 2 3 4 5 6
d) 0 1 2 3 4 5 6

4. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){ int s=0;
 int mas[]={1,2,3,4,5,6};
 for(int i=0; i<4;i++){
  s+mas[i];
        }
 cout<<s<<endl;
 }
```

a) 10
b) 9
c) 11
d) 0

5. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){ int s=0;
 int mas[]={1,2,3,4,5,6};
  for(int i=4; i>=0;i--){
 cout<<mas[i]<<" ";}}
```

a) 10
b) 9
c) 11
d) 0

8. Ko’p o’lchovli massivlar

8.1 Ikki o’lchovli jadvallar

Ko’pincha siz ikki o’lchamli algoritm komponovkasini bor qiymatlarning to’plamini saqlamoqchi bo’lasiz. Bunday ma’lumotlar to’plami moliyaviy va ilmiy ilovalarda uchraydi. Qiymatning ikki qatori va ustunini o’z ichiga oluvchi komponovka ikki o’lchovli jadval yoki matrisa deb nomlanadi. Keling, 11 shaklda

berilgan ma’lumotni qanday saqlashni topamiz: 2010 yil Qishki olimpiyadada fugurali uchish bo’yicha medallar hisobi.

	Олтин	Кумуш	Бронза
Канада	1	0	1
Хитой	1	1	0
Германия	0	0	1
Корея	1	0	0
Япония	0	1	1
Россия	0	1	1
Қўшма Штатлар	1	1	0

8.1.1 Ikki o’lchamli jadvallarni aniqlash

C++ ikki o’lchamli jadvallarni saqlash uchun ikki indiksli jadvaldan foydalanadi. Masalan bu yerda bizning medal hisobi ma’lumotimiz:

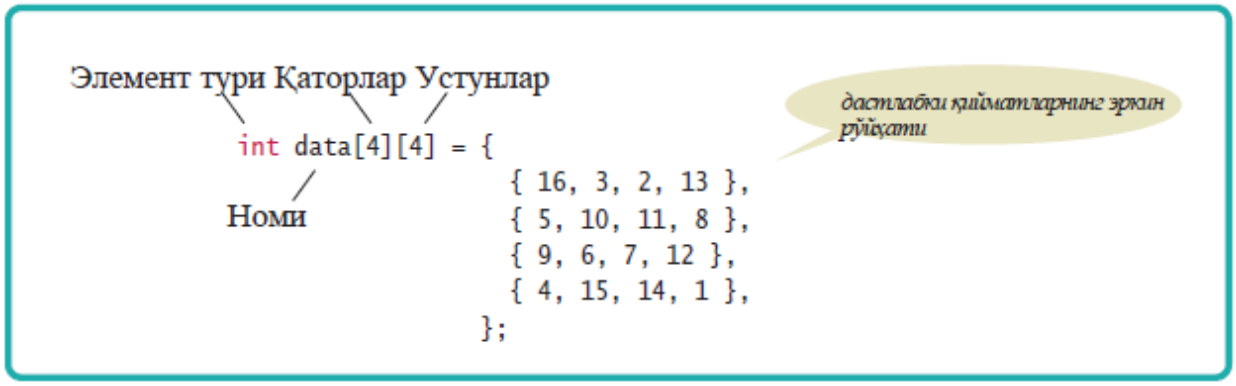
```
const int COUNTRIES = 7;

const int MEDALS = 3;

int counts[COUNTRIES][MEDALS];
```

Siz har bir ustunni quyidagicha guruhlash orqali boshlashingiz mumkin:

```
int counts[COUNTRIES][MEDALS] =

{

{ 1, 0, 1 },

{ 1, 1, 0 },

{ 0, 0, 1 },

{ 1, 0, 0 },

{ 0, 1, 1 },

{ 0, 1, 1 },

{ 1, 1, 0 }
```

```
};
```

Bir o’lchamli jadvalda ham xuddi shunday, ikki o’lchamli jadvalni aniqlanishi bilan hajmini o’zgartira olmaysiz.



8.1.2 Elementlarga kirish imkoni

Ikki o’lchamli jadvalda ma’lum elementga kirish uchun ustun va qatorni nisbiy tanlash uchun alohida diapazonda ikki indiksni aniqlashingiz kerak.

```
int qiymatlar = qiymat[3][1];
```

Ikki o’lchamli jadvalda barcha qiymatlarga kirish uchun qo’yilgan takrorlashlardan foydalanishingizga to’g’ri keladi. Masalan, quyidagi takrorlash hisobning barcha elementlarni chop etadi.

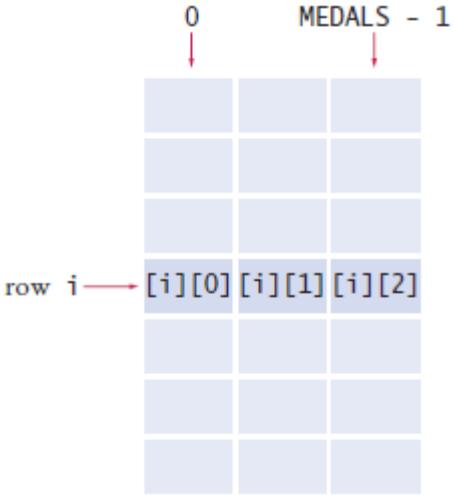
```
for (int i = 0; i < COUNTRIES; i++)
{
for (int j = 0; j < MEDALS; j++)
{
cout << setw(8) << qiymat[i][j];
}
cout << endl; }
```

12 Шакл
икки ўлчамли жадвалда
элементга кириш имкони

8.1.3 Ustun va qatorlarning umumiyligini hisoblash

Ustun va qatorlarning umumiyligini sanash asosiy topshiriq. Quyidagi misolda qatorlar umumiyligi bizga ma’lum davlat tomonida yutib olingan medallarning umumiy sonini beradi.

To’g’ri indeks qiymatini topish biroz mushkul, lekin bu tezkor eskiz yaratish uchun yaxshi fikr. i qator umumiyligini hisoblash uchun quyidagi elementlarni ko’zdan kechirishimiz kerak:

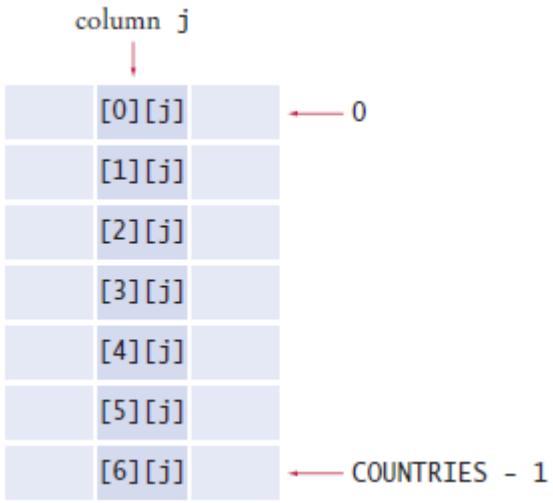


Ko’rib turganingizdek, j 0 dan MEDALS - 1 gacha o’zgarganda[i][j] hisoblarning natijasini hisoblashimiz kerak:

```
int total = 0;

for (int j = 0; j < MEDALS; j++)
```

```
{
total = total + qiymat[i][j];
}
```

ustunlar umumiyligini hisoblash ham shunga o’xshash. i 0 dan COUNTRIES - 1 gacha bo’lganda [i][j] hisoblarining natijasidan:




```
int total = 0;
for (int i = 0; i < COUNTRIES; i++)
{
total = total + counts[i][j];
}
```

8.1.4 Ikki o’lchamli jadval parametrlari

Ikki o’lchamli jadvaldan funksiyaga o’tishda ustunlar sonini parametr turi bilan turg’un koeffisiyent singari aniqlashtirishingiz kerak. Masalan, bu funksiya berilgan qatorni umumiy sonini hisoblaydi:

```
const int COLUMNS = 3;
int row_total(int table[][COLUMNS], int row)
{
```

```
int total = 0;
for (int j = 0; j < COLUMNS; j++)
{
total = total + table[row][j];
}
return total;
}
```

Bu funksiya ikki o'lchamli jadvallarning qatorlarning umumiy sonini qatorlarning erkin soni orqali hisoblashi mumkin, lekin jadval uch ustunga ega bo'lishi kerak. Agar 4 ustunli ikki o'lchamli jadvallarning ustunlarining umumiy sonini hisoblamoqchi bo'lsangiz turli funksiyalar yozishingizga to'g'ri keladi.

Bu limitni tushunish uchun jadval elementlari xotirada qanday saqlanganini bilishingiz kerak. Jadval ikki o'lchamli bo'lib ko'rinsada elementlar bir chiziqli navbat kabi saqlanadi.

Masalan, [3][1]hisobiga erishish uchun dastur 0, 1, va 2 qatorlarini qoldirib o'tishi kerak keyin 3 qatorda offset 1ni joylashtirish kerak. Jadvalning boshidan `offset 3 × ustunlar soni + 1`

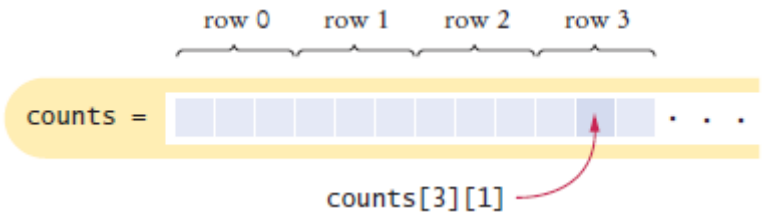
Endi row_total funksiyasini faraz qiling. Translyator elementni topish uchun kodlarni navbatma navbat qo'yadi.

```
table[i][j]
offset ni hisoblash orqali
i * COLUMNS + j
```

Translyator parametrni topish vaqtida ikkinchi juft qavsda ta'minlangan qiymatdan foydalanadi:

```
int row_total(int table[][COLUMNS], int row)
```

Shuni ta'kidlab o'tish kerakki birinchi juft qavs bo'sh bo'lishi kerak, xuddi shunday bir o'lchamli jadvallarda ham.



Qatorlarni umulashtiruvchi funksiyasi vektordagi qatorlar sonini bilishi shart emas. Ag ar qatorlar soni talab etilsa, uni xuddi quyidag i misolda kelt irilg andek uni o’t kazib yuboring:

```
int column_total(int table[][COLUMNS], int rows,
int column)
{
int total = 0;
for (int i = 0; i < rows; i++)
{
total = total + table[i][column];
}
return total;
}
```

Ikki o’lchamli vektorlar bilan ishlash quyidagi dasturda aks ettirilgan. Dastur hisob kitoblarni va qatorlarni chop etadi.

```
#include<iostream>
using namespace std;
const int satr_soni=10;
const int qator_soni=10;

void jadval_kiritish(double
jadval[satr_soni][qator_soni], int elementlar_soni){
```

```
        if(elementlar_soni > satr_soni){return ;}
        for(int i=0; i<elementlar_soni; i++)
        {
           for(int j=0; j<elementlar_soni; j++)
               cin>>jadval[i][j];}
        }

    void jadval_chiqarish(double
jadval[satr_soni][qator_soni], int elementlar_soni){
        if(elementlar_soni > satr_soni){return ;}
        for(int i=0; i<elementlar_soni; i++)
        {
           for(int j=0; j<elementlar_soni; j++)
               {cout<<jadval[i][j]<<" ";} cout<<endl;
        }
    }
    int main(){
        double jadval[satr_soni][qator_soni];
        int elementlar_soni;
        cout<<"Elementlar sinini kiriting: ";
cin>>elementlar_soni; elementlar_soni/=2;
        jadval_kiritish(jadval, elementlar_soni);
        jadval_chiqarish(jadval, elementlar_soni);
    }
```

1. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){ int s=0;
 int mas[10][10];
   mas[0][1]=12;
  cout<<mas[1][0]<<" ";  }
```

a) 10
b) Kompilyatsiya xatoligi
c) 11
d) Musor qiymat chiqaradi

2. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){ int s=0;
 int mas[10][10]; int n; cin>>n;
   for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
        if(mas[i][j]%2==0) s+=mas[i][j];
        } }
        cout<<s<<endl; }
```

a) 10
b) Kompilyatsiya xatoligi
c) 11
d) Musor qiymat chiqaradi

3. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){ int s=0;
 int mas[10][10]; int n=4;
    mas[0][0]=1;    mas[0][1]=2;
    mas[1][0]=3;    mas[1][1]=4;
  for(int i=0;i<2;i++){
        for(int j=0;j<2;j++){
         s+=mas[i][j];       } }
        cout<<s<<endl; }
```

a) 10
b) Kompilyatsiya xatoligi
c) 11
d) Musor qiymat chiqaradi

4. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){ int s=0;
 int mas[10][10]; int n=4;
    mas[0][0]=1;    mas[0][1]=2;
    mas[1][0]=3;    mas[1][1]=4;
  for(int i=0;i<2;i++){
        for(int j=0;j<2;j++){
```

```
        if(mas[i][j]%2==0) s+=mas[i][j];        } }
        cout<<s<<endl; }
```

a) 10
b) 6
c) 11
d) 5

5. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){ int s=0;
 int mas[10][10]; int n=4;
    mas[0][0]=0;    mas[0][1]=2;
    mas[1][0]=3;    mas[1][1]=4;
  for(int i=0;i<2;i++){
        for(int j=0;j<2;j++){
        if(mas[i][j]%2>0 || mas[i][j]%2<0) s+=mas[i][j];   } }
        cout<<s<<endl; }
```

a) 2
b) 4
c) 3
d) 1

6. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){ int s=0;
 int mas[10][10]; int n=4;
    mas[0][0]=0;    mas[0][1]=2;
    mas[1][0]=3;    mas[1][1]=4;
  for(int i=0;i<2;i++){
        for(int j=0;j<2;j++){
        if(mas[i][j]>0 && mas[i][j]<2) s+=mas[i][j]; } }
        cout<<s<<endl; }
```

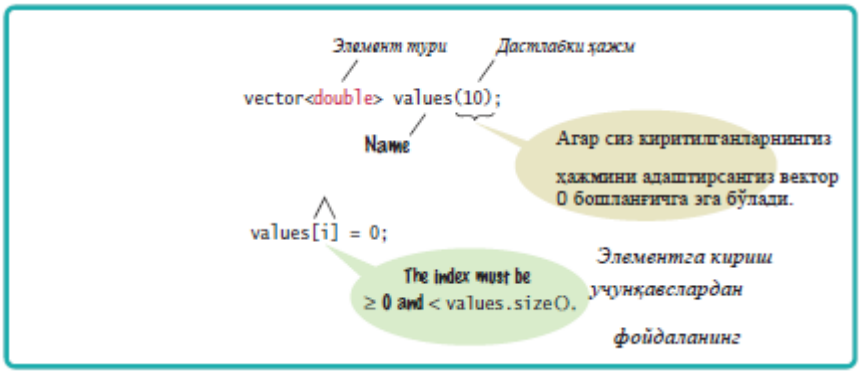
e) 1
f) 2
g) 0
h) 4

8.2 Vektorlar

Foydalanuvchi kiritmasidan qiymatlarni yig'uvchi dasturni yozganingizda nechta qiymatlar bo'lishini bilmaysiz. Afsuski, dastur tuzil ganda jadval hajmi aniq bo'lishi kerak. Biz quyidagi bo'limlarda muhokama qiladigan bu vektor tuzilishi

qulayroq yechimni taklif etadi. Vektor qiymatlar ketma ketligini xuddi jadval qilgandek yig'adi lekin uning hajmi o'zgarishi mumkin.

Vektorni aniqlash:



8.2.1 Vektorlarni aniqlash

Quyidagicha:

```
vector<double> qiymatlar;
```

Boshlang'ich hajmni erkin aniqlashingiz mumkin. Masalan, Bu yerda boshlang'ich hajmi 10 ga teng bo'lgan vektorning ta'rifi:

```
vector<double> qiymatlar(10);
```

agar vektorni boshlang'ich hajmsiz aniqlasangiz uning hajmi 0 ga teng.

Jadvalning 0 hajmini aniqlashda hyech qanday nuqta bo'lmaganda boshlang'ich hajmi 0 ga teng vektorlarga ega bo'lish foydali va keyin ular keraklicha o'stiriladi. Dasturingizda vektorlardan foydalanish uchun siz vektor sarlavhasini ham kiritishingiz zarur.

ector<int> numbers(10);	10 бутун вектор	
vector<string> names(3);	3 тросли вектор	
vector<double> values;	0 ҳажмли вектор	
vector<double> values();	Хато: Векторни аниқламади.	

vector<int> numbers;
for (int i = 1; i <= 10; i++)
{
numbers.push_back(i);
}
	1, 2, 3, ..., 10 билан тўлдирилган бутун векторлар	

vector<int> numbers(10);
for (int i = 0; i < numbers.size(); i++)
{
numbers[i] = i + 1;
}
	10 бутун векторни аниқлашнинг бошқа йўли 1, 2, 3, ..., 10.	

Siz qiymatlar kabi vektor elementlariga kira olasiz, xuddi shunday jadvallar bilan ham.

Hajm a’zosi funksiyasi vektorning joriy hajmini qaytaradi. Barcha vektor elementlari aylanib o’tadigan takrorlashda hajmi a’zosi funksiyasini quyidagicha qo’llang:

```
for (int i = 0; i < qiymatlar.size(); i++)
{
cout << qiymatlar[i] << endl;
}
```

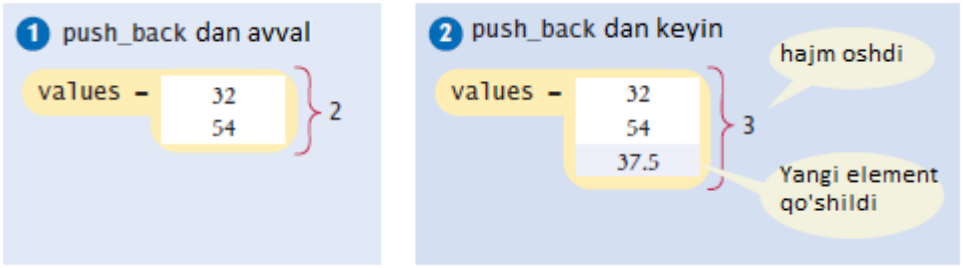
8.2.2 Vektor o’lchamini kattalashtirish va kichraytirish

Agar qo’shimcha elementlar kerak bo’lsa elementni vektorning oxiriga qo’shish uchun **push_back** funksiyasini qo’llaysiz. bunda uning hajmi 1 ga ortadi. **Push_back** funksiyasi quyidagi nuqtali notasiya bilan chaqirishingiz shart bo’lgan a’zo funksiyasidir:

```
qiymatlar.push_back(37.5);
```

Bu chaqiruvdan so’ng 14 shakldagi vektor qiymati 3 hajmga teng va qiymatlar

```
qiymatlar[2] qiymat 37.5 ga teng.
```



Bo’sh vektor bilan boshlash va push_back funksiyasi bilan uni to’ldirish ommalashgan. Masalan,

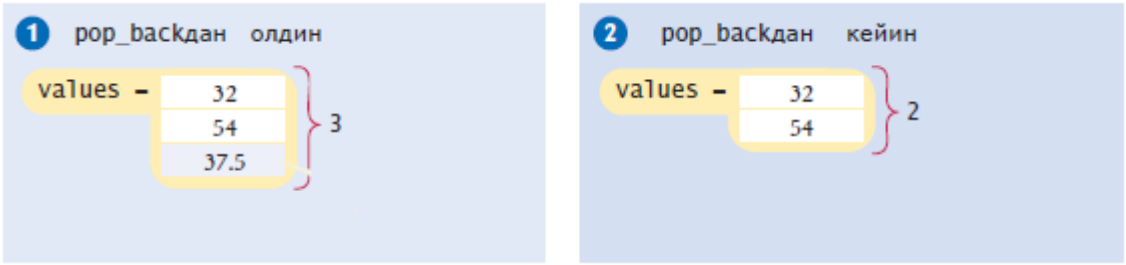
```
vector<double> qiymatlar; // Dastlabki bo’sh

qiymatlar.push_back(32); // endi qiymatla 1 hajmga va element esa 32 hajmga ega

qiymatlar.push_back(54); // endi qiymatlar 2 hajmga va element esa 32,54 hajmga

ega qiymatlar.push_back(37.5); // endi qiymatlar 3 hajmga va element esa

32,54,37.5 hajmga ega
```

push_back a’zo funksiyasi uchun boshqa bir foydalanish bu vektorni kiritma qiymatlari bilan to’ldirish.

```
vector<double> qiymatlar; // Dastlabki bo’sh

double input;

while (cin >> input)

{

qiymatlar.push_back(input);

}
```

Bu kiritma takrorlash 6.2.10 bo’limdagidan osonroq va oddiyroq ekanini qayd eting. Boshqa a’zo funksiyasi, pop_back, vektorning oxirgi elementini olib tashlash, uning xajmini bittaga kichiklashtirish.

```
qiymatlar.pop_back();
```



8.2.3 Vektorlar va funksiyalar

Siz boshqa qiymatlar kabi vektorlarni funksiya parametri sifatida ishlatishingiz mumkin. Masalan, quyida funksiya flouting komponentlik nuqtasi sonlari vektorining umumiy sonini hisoblaydi:

```
double sum(vector<double> qiymatlar)
{
double total = 0;
for (int i = 0; i < qiymatlar.size(); i++)
{
total = total + qiymatlar[i];
}
return total;
}
```

Bu funksiya vektor elementlarini aylanib o’tadi, lekin ularni o’zgartirmaydi.

```
void kopaytirish(vector<double>& qiymatlar, double factor) // Note the &
{
for (int i = 0; i < qiymatlar.size(); i++)
{
qiymatlar[i] = qiymatlar[i] * factor;
```

```
}
}
```

Ba’zi programmistlar o’zgartirilmaydigan vektor parametrlari uchun turg’un yo’nalishdan foydalanadilar.

```
double sum(const vector<double>& qiymatlar) //
const & added for efficiency
```

funksiya vektorni qaytarishi mumkin. Yana vektorlar qolgan qiymatlardan boshqacha emas. funksiyada natijani o’rnating va uni qaytaring. Bu misolda,

```
kvadrat funksiyasi vektorni 02 up to (n – 1)2
qaytaradi:
vector<int> kvadrat(int n)
{
vector<int> result;
for (int i = 0; i < n; i++)
{
result.push_back(i * i);
}
return result;
}
```

Ko’rib turganingizdek funksiyali vektordan foydalanish oson, eslab qolinishi kerak bo’lgan qoidalari yo’q.

8.2.4 Vektor algoritmlari

Ko’pgina algoritmlar o’zgatirishsiz foydalaniladi — qiymat hajmini shunchaki almashtiring. Bu bo’limda turli vektorlarning algoritmlarini muhokama qilamiz.

Ko’chirish

Jadvalning kopiyasini olish uchun aniq takrorlash kerak bo’ladi. Bu vektorning nus’hasini olishdan ko’ra osonroq. Siz oddiygina uni boshqa vektorga qo’yasiz. Masalan:

```
vector<int> kvadrat;
for (int i = 0; i < 5; i++) { kvadrat.push_back(i * i); }
vector<int> lucky_numbers; // Initially empty
lucky_numbers = kvadrat; // Now lucky_numbers contains the same elements as squares
```

O’xshashlikni topish

Ba’zida siz barcha o’xshashlikni topishni hohlaysiz. Bu jadvallar bilan uzoq davom etadi, lekin o’xshashlikni yig’uvchi vektorni qo’llash mumkin. Bu yerda biz 100 dan katta bo’lgan barcha elementlarni yig’dik:

```
vector<double> matches;
for (int i = 0; i < qiymatlar.size(); i++)
{
if (qiymatlar[i] > 100)
{
matches.push_back(qiymatlar[i]);
}
}
```

Elementni olib tashlash

Vektordan biror elementni olib tashlaganingizda pop_back a’zo funksiyasini chaqirish orqali uning hajmini to’g’irlamoqchi bo’lasiz. Bu yerda tartib ahamiyatga ega bo’lmaganda jadval elementini olib tashlash kodi

```
[pos]:
```

```
int last_pos = qiymatlar.size() - 1;
qiymatlar[pos] = qiymatlar[last_pos]; // Replace element at pos with last element
qiymatlar.pop_back(); // Delete last element
```

Buyurilgan vektordan biror elementni olib tashlashda birinchi elementni olib tashlab keyin uning hajmini qisqartiring:

```
for (int i = pos + 1; i < qiymatlar.size(); i++)
{
qiymatlar[i - 1] = qiymatlar[i];
}
qiymatlar.pop_back(); Elementni qo’shish
```

Vektor oxiriga element qo’shish qo’shimcha kod talab etmaydi. push_back a’zo funksiyasidan foydalanishning o’zi kifoya.

Elementni o’rtaga qo’yganingizda vektor hajmini kattalashtirishni xoxlaysiz. Quyidagi koddan foydalaning:

```
int last_pos = qiymatlar.size() - 1;
qiymatlar.push_back(qiymatlar[last_pos]);
for (int i = last_pos; i > pos; i--)
{
qiymatlar[i] = qiymatlar[i - 1];
}
qiymatlar[pos] = new_element;
```

Mavzuga doir test savollari:

1. Quyidagi ifoda qanday natija chiqaradi?

int main(){

```
vector<double> values;
 int i=0,n=3; double qiymat;
while (i<n)
{  cin>>qiymat;
 values.push_back(qiymat); i++;}
for(int i=0;i<n;i++){
        cout<<values[i]<<" ";}}
```

a) 10
b) 3 4 5
c) Qiymat g anima kiritilsa shular chiqadi
d) Musor qiymat chiqaradi

2. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
vector<double> values;
 int i=0,n=3; double qiymat;
while (i<n)
{  cin>>qiymat;
 values.push_back(i+2); i+=2;}
for(int i=0;i<n;i++){
        cout<<values[i]<<" ";}}
```

a) 2 4 6
b) 1 2 3 4
c) Qiymat g anima kiritilsa shular chiqadi
d) 1 3 5 7

3. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
vector<double> values;
 int i=0,n=3;  double qiymat;
while (i<n)
```

```
{  cin>>qiymat;
 values.push_back(i+2); i+=2;}
 values.pop_back();
for(int i=0;i<n-1;i++){
        cout<<values[i]<<" ";}
}
```

a) 2 4 6 8
b) 2 4
c) 2
d) 1 7

4. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
vector<double> values;
 int i=0,n=3;  double qiymat;
while (i<n)
{  cin>>qiymat;
 values.push_back(i+2); i+=2;}
 values.pop_back();
for(int i=0;i<n-1;i++){
        if(i>n)cout<<values[i]<<" ";}
}
```

a) 1 4 3 8
b) 1 5
c) Hech narsa chiqmaydi
d) 0

5. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
vector<double> values;
 int i=0,n=3;  double qiymat;
```

```
while (i<n)
{  cin>>qiymat;
 values.push_back(i); i++;}
 values.pop_back();
for(int i=0;i<n-1;i++){
        if(i%2)cout<<values[i]<<" ";}
}
```

a) 8
b) 1
c) Hech narsa chiqmaydi
d) 33

Nazorat savollari:

1. Agar yig'indi funksiyasini chaqirib uning miqdori haqida yolg'on aytsangiz nima bo'ladi? Masalan, chaqiruv double result = sum(qiymatlar, 1000); xattoki qiymat 100 miqdorga ega bo'lsa ham.
2. Birinchi beshta kvadratlarni hisoblash va natijani massiv sonlarida saqlash uchun kvadratlar funksiyasini qanday chaqirasiz?
3. Massivda elementning birinchi joyini(o'rnini) qaytaruvchi funksiyani yoki agar element yo'q bo'lsa 1ni yozing.
4. Kiritishlarni o'qish funksiyani qaytadan yozingki, massiv miqdori qaytish qiymati emas tayanch parametri bo'lsin.
5. Bir massivga ikkita massivni qo'shadigan funksiya uchun header (nom) yozing. Funksiyani amalga oshirmang.

9. Ko’rsatkichlar

Chap tomondagi o’rinda, spinner ko’rsatkichi element tomon harakatlanmoqda. O’ninchi ko’rsatkich tamonga yo’nalmoqda va elementga ishlov beradi, koptokni qabul qilib yoki yozilgan instruksiyaga rioya qiladi. C++dayam ko’rsatkichlar mavjud, ular har hil ish jarayonini ko’rsatib bera oladi. Ko’rsatkichlar ma’lumotlar bilan ishlashga yordam beradi, joynig o’zgarishi yoki hajm aniqligini hisoblaydi.

9.1 Ko’rsatkichdan foydalanish va uningta’rifi

Ko’rsatkich yordamida, joylashuvi farqlanishi mumkin. Bu hislat ko’p foydali taminga ega. Ko’rsatkichlar yordamida programmani turli qismlarini har hil ma’nolar almashuviga qarab ishlatishi mumkin. Undan tashqari 10 - bobda ko’rishiz mumkinki, ko’rsatkichlar programma tuzishda muhim ahamiyatga ega, bular obektlarni bog’liq turlarini boshqaradi. Bu bobda siz ko’rsatkichdan qanday foydalanishni bilib olasiz.

9.1.1 Ko’rsatkichning ta’rifi

Bank omonatlari va pul o’tqaznalari dasturini yaratadigan dasturchini kurib chiqaylik, bu har doim bir xil bank hisobdan foydalana olmaydi. Ko’rsatkich bilan foydalanib , siz panjara bilan boshqa hisobga o’tishingiz mumkin. Keling o’zgaruvchidan boshlaymiz, ikki barovar **harry_account**= 0

Muvozanat hisobining saqlanishi uchun bankdagi hisobni manipulyatsiya qiladigan algoritm yozmoqchimiz, **harry_account** = 0 dan foydalanib yozishimiz mumkin, ba’zida boshqa hisobdan foydalanishimiz mumkin. Foydalanish ko’rsatkichi bizga shunaqangi egiluvchanlik beryapti. Ko’rsatgichdan bilishimiz mumkinki qiymatning nima ekanligi muhim emas, qayerda turganligi muhimligini. Bu yerda o’zgaruvchan ko’rsatkichni joylashg o’rnini aniqlash pul_miqdoridagi manzil bilan ko’rsatiladi.

Ikkilangan xil yoki "ikkilangan ko’rsatkich", O’zgaruvchining ikkilangan

joylashish mazilini belgilaydi. Operator ham manzil operatori deb nomlanib, o’zgaruvchining manzilini ko’rsatadi. Ikkilangan o’zgaruvchining manzilini olishda qiymat ikki barovar natija beradi*.

Ko’satkich haqida o’ylaganingizda u mavhumdek tuyulishi mumkin, lekin siz undan haqiyqisini qalbakilaridan farqlashda foydalanishingiz mumkin Har bir o’zgaruvchi kompyuter xotirasining ma’lum bir joyida joylashgan bo’ladi. har bir o’zgaruvchi qayerda saqlanayotgani haqida bilmaysiz, qayerda nima bo’lyatganini kuzatib turishingiz mumkin. Bilamizki harrys_pointer 20300da joylashgan. 1 - rasmda ko’rishimiz mumkin, pul_miqdori 0 qiymatga teng, lekin qiymat &pul_miqdori 20300da. Hisoblagichning qiymat ko’satgichi ham 20300da. Diagrammalarda, ko’rsatkich boshlangan joydan boshlab chizib boramiz, lekin kompyuterda chizishni o’rniga, sonlardan foydalanamiz. Ko’rsatkichdan foydalanishda , siz istalgan vaqtda boshqa hisobga o’ta olasiz. Boshqa hisobga o’tishda siz shunchaki uning ko’rsatkichini o’zgartirasiz:

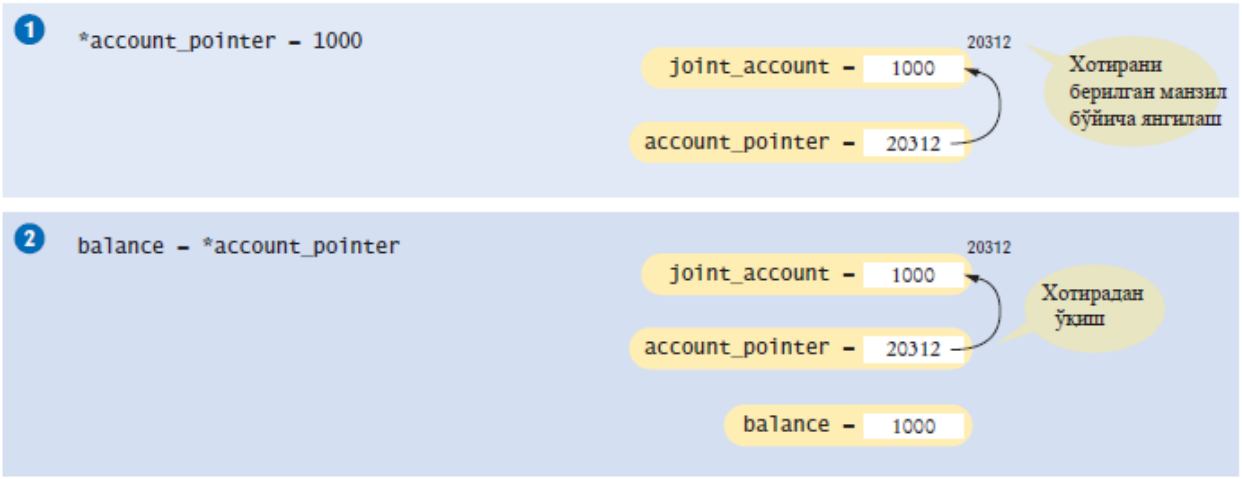
account_pointer = &joint_account;

9.1.2 O’zgaruvchiga ko’rsatkich orqali kirish

O’zgaruvchiga kirish huquqini ko’rsatadi qachonki u bor bo’lsa. Operatorlar o’zgaruvchining ko’rsatkichlarini o’qishda yoki yangilashda foydalanadi. Operatorlarning hyech qanday amallar bilan aloqasi yo’qligida, ko’rsatkichlardan birgalikda foydalanadilar. Standart C++, bu operator Indirection Operator deb nomlanadi lekin umumiy qilib dereferencing operator deb ham nomlasa bo’ladi. **W account** pointer qaysi hisob raqamni ko’rsatsa ariza shunga beriladi.

```
*account_pointer = 1000;
```

Boshqacha qilib aytganda * **account_pointer**dan foydalanish yo’li **harry_account** yoki **jory_account** yo’li bilan judayam o’xshash.



Bunday hisobda ko'rsatuvchi topshiriqning chap yoki o'ng tomonda bo'lishini ifodalaydi. U chap tomonda sodir bo'lsa, o'ng tomondagi qiymati joyida saqlanadi. Qachonki u o'ng tomonda paydo bo'lsa, qiymati joyidan olinadi va chapdagi o'zgaruvchilarga qo'yiladi. Misol uchun, quyidagi bayonot o'zgaruvchilarni o'qiydi va **account_pointer** hisoblaydi, o'zgaruvchilarnig o'z mazmunini joylashtiradi. **balance** = **account_pointer**;

Sizda har ikala tarafda ham account_pointeri bo'lishi mumkin. Quyidagi xabar $100 aniqlaydi

```
*account_pointer = *account_pointer - 100;
```

9.1.3 Boshlangich ko'rsatkichlar

Ko'rsatkichlar yordami bilan, to'g'ri o'rnata olishga e'tibor berish juda muhim sanaladi. Boshlang'ich ko'rsatkichlarda, Ko'rsatkich va xotira manzili bir xil bo'lishiga e'tibor bering

```
int balance = 1000;

double* account_pointer = &balance; // Xatolik!
```

Manzil va balans ko'rsatkichli qiymatga ega bo'ladi. Bu hyech qachon int bilan bir juft ko'rsatkich huquqiy emasligini ko'rsatadi.

Agar siz bir ko'rsatkich o'zgaruvchisini qo'shimcha o'zgaruvchiga yetkazib bermasdan aniqlasangiz, ko'rsatkich tasodifiy manzilni o'z ichiga oladi. Bu tasodifiy manzildan foydalanish xato hisoblanadi. Amalda, dasturingiz sirli

ishlashga o’tadi yoki to’qnashadi agar boshlang’ichsiz foydalansangiz.

```
double* account_pointer;
*account_pointer = 1000;
```

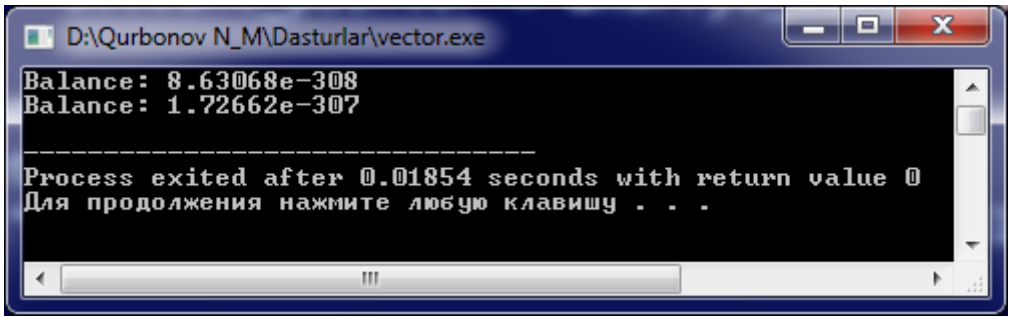
Siz ko’rsatkichni tasvirlashda ishlatishingiz kerak bo’lgan maxsus qiymat NULL hyech qayerda belgilanmagan. Agar siz ko’rsatkich o’zgaruvchisini aniqlasangiz va ular boshlashga tayyor bo’lmasa , NULL o’rnatishga biroz ertalik qiladi.

```
double* account_pointer = NULL; // keyinroq o’rnatiladi
```

Malumotlarga NULL ko’rsatkichi orqali ulanishga harakat qilish noqonuniydir, u sizning dasturingizni bekor qilinishiga sabab bo’lishi mumkin Quyidagi dastur ko’rsatkichning xatti harakatini aniqlab beradi. Biz bir hil qaytarishni amalga oshiramiz ikki marta, lekin hisob ko’rsatkichi uchun turi xil qiymatlar bilan bir hil hisob o’zgartiriladi.

```
#include<iostream>
using namespace std;
int main(){
    double harrys_account = 0;
    double joint_account = 2000;
    double* account_pointer = &harrys_account;

    *account_pointer = 1000;
    account_pointer = account_pointer - 100;
    cout<<"Balance: "<< *account_pointer<<endl;

    account_pointer =&joint_account;
    account_pointer = account_pointer - 100;
```

```
    cout<<"Balance: "<< *account_pointer<<endl;

    return 0;

}
```



Mavzuga doir test savollari:

1. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
int n = 6, A[5] = {0, 1, 2, 3, 4};
int *p; p=A;
cout<<p[3]<<endl;
}
```

a) 1
b) 2
c) 3
d) 4

2. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
int n = 6, A[5] = {0, 1, 2, 3, 4};
int *p; p=&n;
cout<<*p<<endl; }
```

a) 1
b) 2
c) 3
d) 4

3. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
   char a;
   char *p=new char;
   *p='b';
   a=*p;
cout<<a<<endl;
}
```

a) 1

b) a
c) b
d) 4

4. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
   char a='c';
   char *p=new char;
   p=&a;
   a=*p;
cout<<*p<<endl;
}
```

a) d
b) c
c) b
d) a

5. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
   char a;
   char *p=new char;
   p[0]='q';
   a=*p;
cout<<*p<<endl;
}
```

a) a
b) q
c) 3
d) f

9.2 Massiv va ko'rsatkichlar

Ko'rsatkichlar massivlarning o'ziga xosligini tushunish uchun zarur. Keyingi bo'limlarda massiv va ko'rsatkichlar C++ da bir biriga qanday bog'liqligini ko'rsatamiz.

9.2.1 Massivlar ko'rsatkichlar o'rnida

Massivning bayonotini oling:

```
int a[10];
```

Bilganingizdek, a[3] massivni bildiradi. Qavssiz massivning nomi

ko’rsatkichning boshlang’ich elementini bildiradi (3-tasvirnga qarang). Ko’rsatkichni o’zgaruvchining ichida tasvirga olishingiz mumkin:

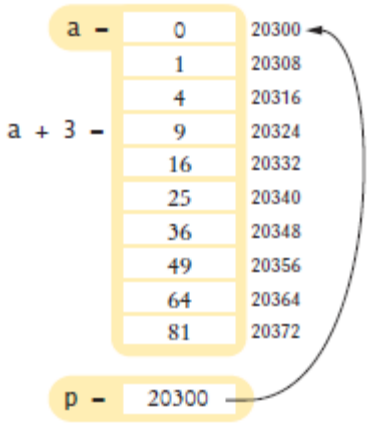
```
int* p = a; // Now p points to a[0]
```

Shuningde, ko’rsatkichning o’rniga massivning nomini ishlatishingiz mumkin.

```
cout << *a;
```

Bu tasdiqlash pastdagi bilan bir xil ta’sirga ega

```
cout << *a;
```



9.2.2 Arifmetik ko’rsatkich

Massivlarga o’zgargan ko’rsatkichlar arifmetik ko’rsatkichlarni qo’llashadi.

Boshqa massiv joylashuvini topish uchun ko’rsatkichga butun son qo’shishingiz mumkin. Masalan, p massivning boshlanishini ko’rsatadi:

```
double a[10];
double* p = a;
```

So’ng

```
p + 3
```

p + 3 i fodasi 3 indeksli massivning ko’rsatkichidir, va

```
*(p + 3)
```

massiv 3 - elementidir.

Oldingi bo'limda ko'rganingizdek, massiv nomini ko'rsatkichday ishlatishimiz mumkin. Bu degani, a + 3 3 indeksli massivning ko'rsatkichidir , va *(a + 3) a[3] bilan bir xil ma'noga ega.

Umuman olganda, har bir butun son n ga

```
a[n] is the same as *(a + n)
```

Bu qonun C++ da hamma massivlar nimaga 0 indeks bilan boshlanishini tushuntiradi.

Ko'rsakich a yoki (a + 0) massivning boshlanish elementini ko'rsatadi. Shuning uchun bu element a[0] bo' li shi kerak. Arifmetik ko'rsatkichni yaxshiroq tushunish uchun, haqiqiy xotira manzilini bilamiz deb tasavvur qilaylik. Tasavvur qiling, a array 20300 manzilida boshlanadi. Massiv double turidagi 10 ta baxoni o'z ichiga oladi. Double baxosi 8 bayt xotirani egallaydi. Shuning uchun massiv 80 bayt xotirani egallaydi, 20300 dan 20379 gacha. Boshlang'ich baxo 20300 manzilida joylashgan, keyingisi 20308 manzilda, va h.k. Masalan, a + 3ning baxosi 20300 + 3 x 8 =20324.

317-betdagi 3-jadval arifmetik ko'rsatkich va massiv/ko'rsatkichning ikki taraflama bog'lanishini shu misol orqali ko'rsatadi.

9.2.3 Ko'rsatkichlar - massivning o'zgaruvchanligi

Agar massiv va ko'rsatkichlar o'rtasidagi bog'liqlikni tushungan bo'lsangiz, massiv parametrini boshqa turdagi parametrlardan farqini bilasiz.Misol tarzida, massivning hamma baxolarini yig'indisini hisoblaydigan funksiyani oling:

```
double sum(double a[], int size)
{
double total = 0;
for (int i = 0; i < size; i++)
{
```

```
total = total + a[i];
}
return total;
}
```

Bu yerda funksiya berilgan(4-tasvirni ko’ring):

```
double data[10];
... // data ni insalizatsiya qilish
double s = sum(data, 10);
```

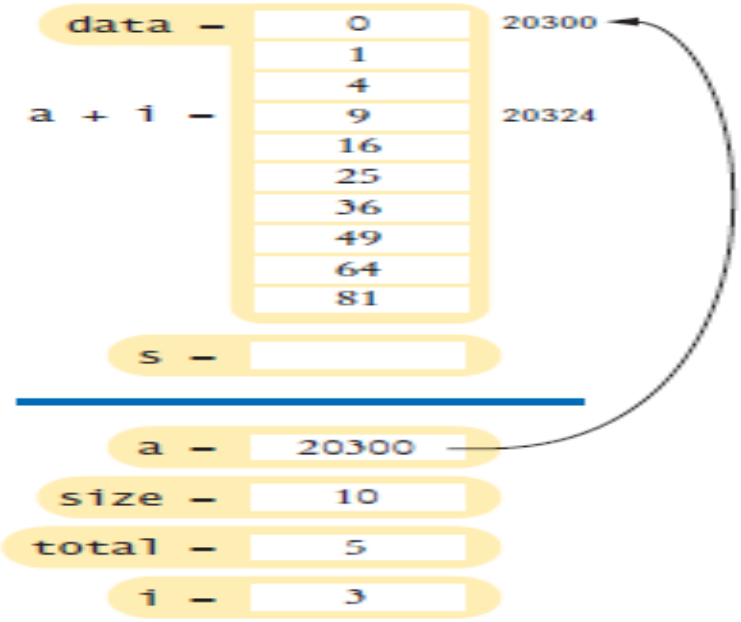
Baxoning ma’lumoti funksiyaning yig’indisida berilgan. u odatda double* turidagi ko’rsatkich, massivning boshlang’ich elementini ko’rsatadi. Shuning uchun funksiya shu tarzda bo’ladi deb kutiladi

```
double sum(double* a, int size)
```

Biroq, funksiyaga yaxshilab qarasangiz,parametrning o’zgaruvchanligi massivning bo’sh chegaralari bilan e’lon qilinishini ko’rasiz :

```
double sum(double a[], int size)
```

C++ ning tuzuvchisiga ko’ra, bu parametrlar mutlaqo bir xil. [] bu belgi - ko’rsatkichni e’lon qiladigan. Kompyuter olimlari bu terminni odamlar o’qishi uchun oson va mukammal tafsilotlarni yashirish uchun ishlatishadi. Massiv funksiyaga faqat boshlang’ich manzil beriladi.



Massivdan o’tish uchun ko’rsatkichni ishlatish:

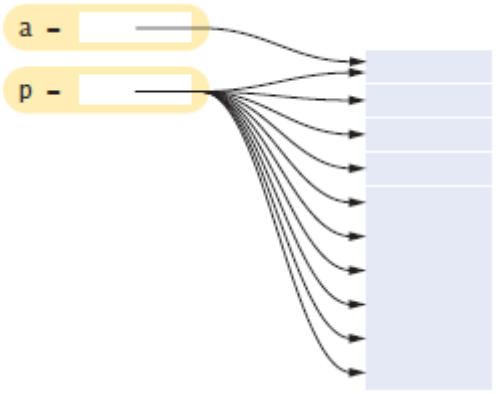
Endi bilganingizday, funksiya yig’indising birinchi parametri-bu ko’rsatkich, funksiyani biroz boshqacha tarzda ifoda qilishingiz mumkin. Butun son indeksini ko’paytirish o’rniga, massiv elementlarini ko’rish uchun ko’rsatkich o’zgaruvchanligini ko’paytirishingiz mumkin:

```
double sum(double* a, int size)
{
double total = 0;
double* p = a;
for (int i = 0; i < size; i++)
{
total = total + *p;
p++; }
return total;
}
```

Aslida, p ko’rsatkichi a[0] elementini ko’rsatadi.

p++; uni keyingi elementga olib boradi.

Massiv elementini a[i] qilib qabul qilgandan ko’ra, uni ko’paytirgan maqul. Shu sababdan, ayrim programmistlar massiv elementlariga yetish uchun indekslardan emas, ko’rsatkichlardan foydalanishadi.



Aniq programmalashtiring:

Ayrim programmistlar instruksiyalar sonini kamaytirishga urunishadi, natija kodi tushunish uchun qiyin bo’lsa ham. Masalan, bu yerda funksiya yig’indisining to’g’ri joriy etilishi berilgan:

```
double sum(double* a, int size)
{
double total = 0;
while (size-- > 0) {
total = total + *a++; }
return total;
}
```

Bu 2 yo’l bilan amalga oshirildi. Birinchisi, funksiyaning parametri va o’lchami o’zgaruvchan, qiymatlarni o’zgartirish-qonuniy. Ifodalarning o’lchami -- va ++ o’zgaruvchanning ko’payishi yoki kamayishi va eski baxosiga qaytishini anglatadi. Boshqa so’z bilan aytganda **size-- > 0** ifodasi 2 ta vazifani bajaradi: o’lchamni kamaytirish va undan oldin uni pozitivligini tekshirish. Huddi shunday ***a++** ko’rsatkichni keyingi elementga ko’paytiradi, va elementni ko’payishdan oldingi xoliga qaytaradi.

Iltimos bunday programmalash uslubini ishlatmang. Sizni ishingiz o’z aqlingiz bilan boshqalarni ko’zini aamashtirish emas, balki tushunish oson bo’lgan kodni yozish.

Ko’rsatkichni o’z joyiga qaytarish:

2 ta elementli massivga ko’rsatkichni qaytarmoqchi bo’lgan funksiyani oling, massivning birinchi va so’nggi baxolari:

```
double* firstlast(double a[], int size)
{
double result[2];
result[0] = a[0];
result[1] = a[size - 1];
return result; // Xatolik!
}
```

Funksiya ko’rsatkichni massivning boshlang’ich elementiga qaytardi. Biroq, bu massiv birinchi funksiyaning o’zgaruvchanidir. Funksiya chiqib ketsa o’zgaruvchan ham ketadi. Yaqinda ular boshqa funkssiyalar bilan to’ldiriladi. Massivni kesib o’tib mu masalani yechishingiz mumkin:

```
void firstlast(const double a[], int size, double[] result)
{
result[0] = a[0];
result[1] = a[size - 1];
}
```

O’zgarmas ko’rsatkichlar:

Quyidagi ifoda doimiy ko’rsatkichni aniqlaydi:

```
const double* p = &balance;
```

r ko’rsatkichdagi bahoni o’lchay olmaysiz. Quyidagi ifoda noto’g’ri:

```
*p = 0; // xatolik
```

Albatta, baxoni qisangiz bo’ladi:

```
cout << *p; // to’gri
```

Doimiy massivning parametr o’zgaruvchanligi doimiy ko’rsatkichga teng. Masalan:

```
double sum(const double[] values, int size)
```

Funksiya shunday aniqlanishi mumkin

```
double sum(const double* values, int size)
```

Massiv elementlarini o’qish uchun funksiya ko’rsatkich baxolarini ishlatadi, ammo ularni aniqlayolmaydi.

9.3 Belgili massivlar

"harrys" ga o’xshagan torlar konstanta hisoblanadi. Uning belgilarini

aniqlash mumkin emas. Agar tordagi belgilarni aniqlamoqchi bo’lsangiz, massiv belgisini hisoblang. Masalan:

```
char char_array[] = "Harry";
```

Bu char_array massiv o’zgaruvchi 6 yacheykadan iborat, ‘H’, ‘a’, ‘r’, ‘r’, ‘y’ so’zi bilan va bir null qiymati bilan to’ldiriladi. Komplyator berilgan ma’lumotni korsatkichga taqsimlaydi va oxiriga NULL qo’yib chegaralaydi. Siz massivdagi ma’lumotlarni keyinchalik o’zgartirishingiz mumkin:

```
char_array[0] = 'L';
```

9.3.1 Qatorlarni C va C++ ga o’girish

Oldin, belgi ketma-ketliklar to’g’ridan-to’g’ri manipulyatsiyasi qilindi, so’ngra esa C ++ satring sinfi keng foydalanish mumkin bo’ldi. Siz C++ string sinfini yuboruvchi yoki qabul qiluvchi funksiyalardan foydalansiz, bunda C

qatorlari va qator obektlari qanday o’zgartirilishini bilishingiz shart. Masalan

```
int atoi(const char s[])
```

```
char[] year = "2012";
```

```
int y = atoi(year); // Now y is the integer 2012
```

Masalan ,<cstdlib> kutubxonasi foydali funksiyalarni e’lon qiladi.

int atoi(count **char** s[]) bu atoi funksiya satr tipli sonlarni int tipli songa o’zgartiradi

```
string year = "2012";
```

```
int y = atoi(year.c_str());
```

Bu funkwiyalar **String** sinfida yetarli emas. String sinfi castr a’zosi funksiya "berilgan elementni" chaqiradi. Agar String tipida bo’lsa, so’ngra s.c_str() char* pointer yoziladi.

string name = "Harry"; C++ string sinfini "Harry" so’zi bilan to’ldirilyapti

Aksincha, S qatoridan C ++ qatorida o’tkazish juda oson. string o’zgaruvchisini qiymat bilan ta’minlash uchun oddiygina qilib char tipidagi ko’rsatkich yoki massiv qiymati tenglab beriladi.

9.3.2 C++ strings sinfi va [] operatorlar

Bu bo’limda biz hamisha string sinfini alohida elementiga murojaat etish uchun substr funksiyasidan foydalanamiz.Masalan: Agar qator quyidagicha berilgan bo’lsa:

```
string name = "Harry";
```

```
bu ifoda
```

```
name.substr(3, 1)
```

bu qator 3 indeksda joylashgan 1 ta elementni qaytaradi. Shuningdek siz alohida elementlariga mirojat uchun [ ] operatoridan foydalanishngiz mumkin:

```
name[0] = 'L';
```

Yendi qator sifatida "Larry" dan foydalanadi. [ ] operatori substr funksiyasiga nisbatan juda qulay hisoblanadi. Masalan. quidagi qator barcha elementlarni o’zlashtiradi va qiymatni butunlay o’zgartiradi.

```
/**
Sartni kattalashtirish.
@param str a string
@return a string o’z holiga qaytaradi
*/
string uppercase(string str)
{
string result = str; // Make a copy of str
for (int i = 0; i < result.length(); i++)
{
result[i] = toupper(result[i]); // Natijalarni
jamlaydi
}
return result;
}
```

C strings bilan ishlash:

String guruhi keng tarqalishidan oldin,massiv belgilari bilan to’g’ri ishlash oson edi. C stringlarini boshqaradigan bir nechta funksiyalar berilgan. Hamma funksiyalar <s_string>ni boshqaruvida ko’rsatilgan. Ism va familiyani stringda bir-biriga bog’lang. String guruhi buni oson bajaradi:

```
string first = "Harry";
string last = "Smith";
string name = first + " " + last;
```

Keling shu mashqni C stringlar bilan bajaramiz. Massiv belgilarini natijaga joylang.

```
const int NAME_SIZE = 40;

char name[NAME_SIZE];
```

Bu massiv o’z ichiga 39 uzunligidagi torlarni oladi, chunki bitta belgi nol terminatori uchun. Yendi ismingizni strncpy ni ishlatib ko’chiring:

```
strncpy(name, first, NAME_SIZE - 1);
```

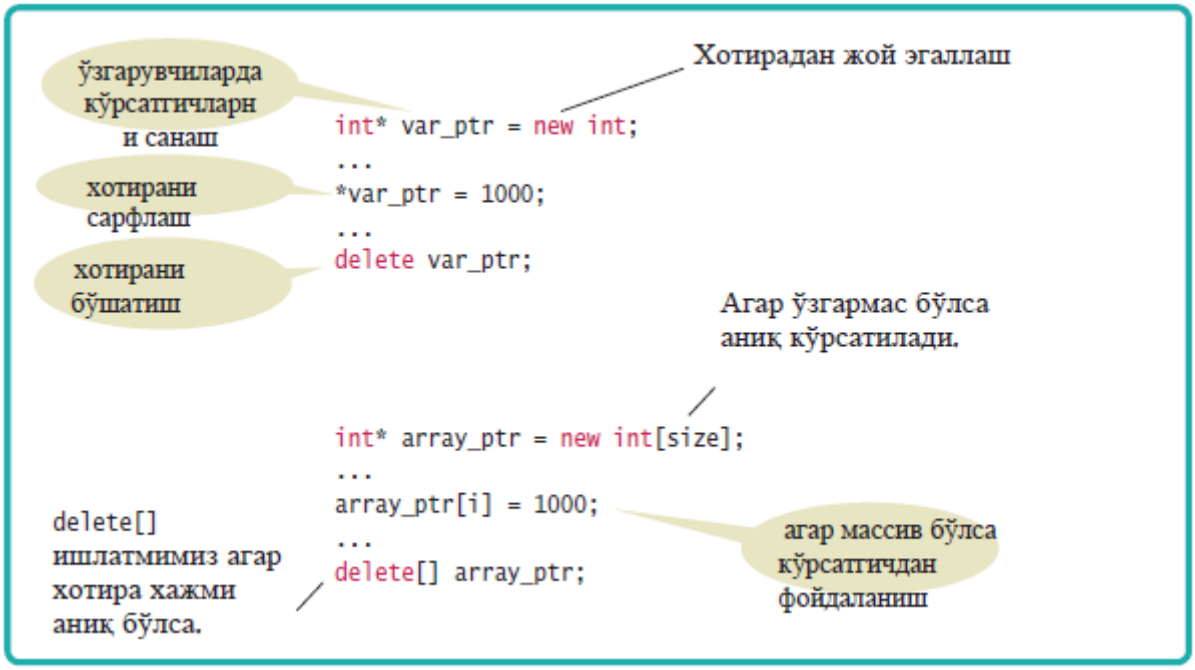
Massiv mo’ljali toshib ketmasligidan ogoh bo’ling. Ism 39 ta belgidan ko’proq bo’lishi mumkin emas, lekin haker xotirani to’ldirish uchun ko’proq joy qoldirgan bo’lishi mumkin. Endi, xona, bo’sh joy va familiya bo’lsa, yana bir bor toshib ketmaslikdan ogoh bo’ling.

```
int length = strlen(name);

if (length < NAME_SIZE - 1)

{

strcat(name, " ");

int n = NAME_SIZE - 2 - length; // Leave room for space, null terminator

if (n > 0)

{

strncat(name, last, n);

}

}
```

Ko’rib turganingizdek, C string C++ dan 3 karra uzunroq, va bu layoqatli emas — massivning mo’ljali yetarli darajada uzun bo’lmasa, u olib tashlanadi.

9.4 Dinamik xotirani taqsimlash

Ko’pchilik programmalashtirish ishlarida, siz oldindan qancha soha kerakligini bilmaysiz. Bu muammoni hal qilish uchun, siz dinamik taqsimlanishdan foydalana olasiz va sizga kerak bo’lgan yangi sohani yaratish uchun C++ da bajarilayotgan amalga murojaat qilasiz. Bajarilayotgan sistema sohalar va massivlarning biror turini taqsimlay oladigan heap deb ataluvchi katta maydonlarni o’zida saqlaydi.



Xotira blokini o’chirgandan keyin, siz undan ko’p foydalana olasiz. Xotira fazosi allaqachon biror joyda foydalanilgan bo’lishi mumkin

```
delete[] account_array;
```

```
account_array[0] = 1000; // Yo’q! siz ko’p bo’lmagan xotiraning account_array ini boshqara olasiz.
```

```
double* bigger_array = new double[2 * n];
for (int i = 0; i < n; i++)
{
```

```
bigger_array[i] = account_array[i];
}
delete[] account_array;
account_array = bigger_array;
n = 2 * n;
```

Bunda vektor aynan nimaga ko'rir orqasida bo'ladi. Hyeap juda ku taqsimlash hisoblanadi, lekin aniq belgila tekshirishda ehtiyot bo'lishingiz kerak:

Nyew uchun har bir chaqiruv o'chirilgan nyew t solshtirilishi kerak.

Massivlarni o'chirishda delete[] foydalaning Memory block o'chgandan keyin unga yozmanng.

Ko'rsatkichlarning osilib qolishi

Juda ko'p qilinadigan ko'rsatkich xato oldin o'chirilgan xotirani ko'rsatishda ko'rsatkichdan foydalanishdir. Ya'nikim osilish ko'rsatkichi deb ataladigan ko'rsatkich. Chunki bo'sh xotira boshqa maqsadlarda foydalaniladi, siz osilish ko'rsatkichi yordamida uni yaratasiz. Ushbu misolni tahlil qiling

```
int* values = new int[n];
delete[] values;
values[0] = 42;
```

Bu kod ko'rsatkich asosiy xotira o'rnini ko'rsatadimi yo'g'ligini kompilyator maydon hosil qilgunicha o'qiydi. Har holda, Dastur kutilmagan natijalar bilan ishlaydi. Agar programma o'chirilgan sohadan keyin biror qismda nyew operatorini chaqirsa, chaqiruv bir xil xotirani taqsimlaydi. Hozir programmaning ba'zi qismlari sohaning ko'rsatkichini xotiraga joylashtiradi, programmani xotirga yozganizda bu programma qism ishi bajariladi. Bu siz xatto buni ko'ra olmasayiz ham sodir bo'ladi—chaqiruv kutunxonada sodir bo'ladi. Ko'rsatkich o'chirilmaganda undan foydalanmang. Ba'zi programmistlar buni NULL orqali tozalashadi:

```
delete[] values;

values = NULL;
```

Bu mukammal himoya emas — siz boshqa ko’rsatkich o’zgaruvchilarga buni saqlasangiz bo’ladi — lekin bu oqilona yo’l emas.

Xotira oqimlari

Boshqa qilinadigan umumiy xato hyeapda xotirani taqsimlanganda. Taqsimlanmaydiga xotira bloki Xotira oqimi deb ataladi. Siz kichik xotira bo’laklarini taqsimlaganda ularni yoki ularni taqsimlamagnizda, siz katta muammoga duch kelmaysiz. Dastur mavjud bo’lganda, barcha taqsimlangan xotira Operatsion sistemaga yuklanadi. Lekin sizning programmangiz uzoq vaqt ishlasa, yoki juda ko’p xotira tasimlansa (yehtimol halqada), keyin u xotirani egalab oladi. Xotira to’ldirilishi programmangiz xatoligiga sabab bo’ladi. Yeng yuqori holatlar, Sizning programmangiz butun xotiraniyegallab olsa Kompyuter qotip qoladi. Xotira oqimidan ehtiyot bo’lish chunki u qayta yuklanmagan sistemalarda oylab yillab ishlaydi. Hatto siz qisqa programma yozsangiz ham, siz xotira oqimini yaratishga asos yaratasiz. Har doim buni oldini olishga harakat qiling.

9.5 Ko’rsatkichli massiv va vektorlar

Siz ko’rsatkichlar ketma-ketligiga ega bo’lsangiz, ularni massiv yoki vektorga joylashtira olasiz. Quyida vektor va ko’rsatkichlarning 10 tasi e’lon qilingan

```
int* pointer_array[10];

vector<int*> pointer_vector(10);
```

so’zlar ketma ketlik bilan indeks i ni ifodalaydi. Turli uzunlikka ega bo’lgan qator quyidagi ko’rsatkich ketma - ketligidagi bir ilovada 2 o’lchamli massiv hisoblanadi.

9.5.1 Ko’rsatkichlar bilan ishlash

Siz Ko’rsatkichlardan foydalanasiz chunki siz progrmmangiz moslanuvchan bo’lishini xohlaysiz: bu esa ma’lumotlar o’rtasida yaqinlikni o’zgartirish qobilyati hisoblanadi.

Qanday qilib ko’rsatkichlar orqali muammolar yechiladi.

Biz rasm nusxalovchini mantiqiy simulatsiylovchi qismini misol qilishimiz mumkin. Odam foydaanuvchi nomerini kiritish bilan nusxa qila oladi. 0 dan 99 gacha bo’lgan 100 ta foydalanuvchi bor. Har bir hisob asosiy hisob yoki qo’shimcha hisob bo’luvchi nusxa hisob bn bog’langan. Bu bog’lanishlar administrator tomonidan tuziladi. Nusxa qilinganda o’zlashtiruvchi hisob buni qabul qiladi.

9.5.2 Strukturalar bilan ko’rsatkich a’zolari

Structuraning azosi ko’rsatkich bo’lishi mumkin. Bu holat odatda malumot structura qiymatlari orasida bolinganda vujudga keladi. Quyidagi misolni hisoblang. Tashkilotlarning ko’p sonli offislarida har bir ishchida ism va ish joylari bor.

```
struct Employee
{
string name;
StreetAddress* office;
}
```

Bu yerda biz ikki ishchining hisoblash idoralarini tariflaymiz:

```
StreetAddress accounting;
accounting.house_number = "1729";
accounting.street_name = "Park Avenue";
Employee harry;
```

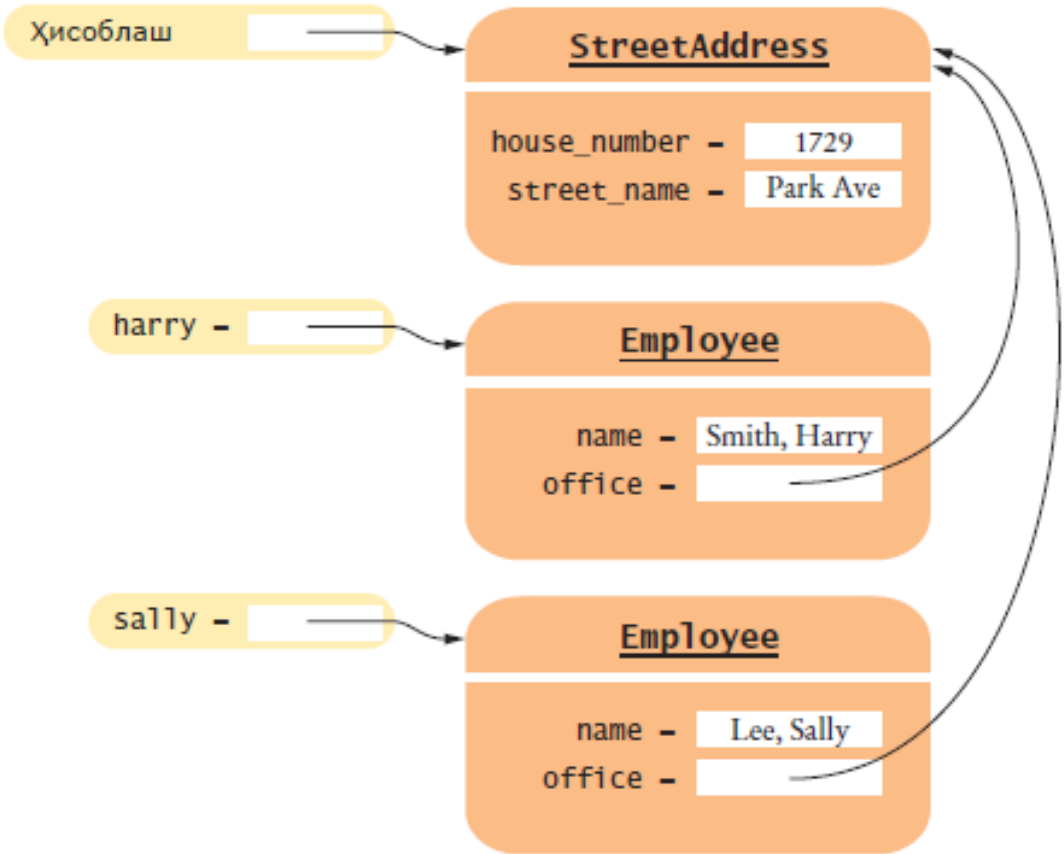
```
harry.name = "Smith, Harry";
harry.office = &accounting;
Employee sally;
sally.name = "Lee, Sally";
sally.office = &accounting;
```

12-raqam structuralarning qanday aloqadorligini ko’rsatadi.

Bu malumotlarni bo’lishishda muhim foyda bor. Faraz qiling isoblash offisiga ko’cha boylab harakat qilinadi:

```
accounting.house_number = 1720;
```

Hozir Sally va Harryning offis manzillari avtomatik ravishda malumotlardan xabardor qiladi.



Nazorat savollari:

1. Ko’rsat nima?
2. Ko’rsatgichlarni parameter sifatida ishlatib bo’ladimi?
3. Dinamik massiv nima?
4. Dinamik xotira qanday ajratiladi?
5. Sinflarda ko’rsatgichli medodlar qandey e’lon qilinadi?
6. Ko’rsatgichli metodlarga qandey murojat qilinadi?
7. O’zgaruvchi joylashgan adres qanday o’zlashtiriladi?
8. Adres olish belgisini ko’rsating va uni yana qandey maqsadlarda ishlatsa bo’ladi?
9. Ko’rsatgichlar qandey masalalarda ishlatiladi?
10. Dinamik massivlar bilan static massivlar farqini nimada?

10. Fayillar va fayillar bilan ishlash

Fayllarni o'qish va yoza olish tizim oqimlaridan foydalanilgan holda satrlar va sonlar orasida o'zgartirish, buyruqning dalillarni qayta ishlash, davriy va tasodifiy kirish tushunchalarni tushunish.

Bu bobda siz, haqiqiy dunyo ma'lumotlarni qayta ishlash uchun C ++ oqimi kutubxona-juda foydali mahorat yordamida fayllarni o'qish va yozishni bilib olasiz. Ilova sifatida, siz ma'lumotlarni qanday shifrlashni bilib olasiz. (Har tarafda ko'rsatilgan Enigma mashinasi Ikkinchi jahon urushida Germaniya tomonidan ishlatiladigan shifrlash qurilmasi. kashshof Britaniya kompyuter olimlar kodni sindirdi va kodlangan xabarlarni to'xtatishga erishdilar, qaysiki urushda g'alaba qozonishga muhim bo'lgan.) Keyinchalik bobda, siz bunday tasvir ma'lumotlarni saqlash o'sha sifatida o'zaro fayllar, qayta ishlashni o'rganasiz.

10.1 Matn fayllarini o'qish va yozish

C++ *input/output* kutubxonasi oqimlar tushunchasiga bog'langan. *Input stream* bu ma'lumotlar manbayi, va *output stream* ma'lumotlar uchun joy. Ma'lumotlar uchun eng keng tarqalgan manbalar va belgilangan joylar sizning qattiq diskingizdagi fayllardir. Data faqat bir ko'chirish tasmasini, bir vaqtning o'zida bir-bandlari kabi kiritish oqimini yetib kelishi ko'rsatilgan

Faylga kirish uchun siz file stream dan foydalanasz. Fayl stream ning asosan 3 turi mavjud: ifstream (input uchun), ofstream (output uchun), va fstream (ikkala input va output uchun). Qachonki siz bu fayllardan birontasidan foydalanganingizda u <fstream> header ni o'z ichiga oladi.

Quyidagi bo'limlarda, siz fayllarni ma'lumotlarni qanday qayta ishlashni o'rganasiz. Bu fayllar katta ma'lumotlar silsilasini tahlil qilishda juda keng tarqalgan, chunki fayl ishlash bir qancha fanlar juda foydali qobiliyat hisoblanadi.

10.1.1 Oqimni ochish

File stream dan biror narsani o'qish uchun uni ochish kerak.Stream ni

ochganingizda diskda saqlangan fayl nomini berasiz. Aytaylik siz programmadek bir xil katalogda joylashgan input deb nomlangan fayldan ma'lumot o'qimoqchisiz. So'ngra fayni ochish uchun quyidagi funksiyalarni bajarasiz:

```
in_file.open("input.dat");
```

Bu bayonot input.dat nomli fayl bilan turli fayllar orasida bog'lanad. Barcha stream lar obyektlardir, va siz ularni manipulyatsiya qilish uchun dot notation funksiyasidan foydalanishingiz mumkin. Yozish uchun faylni ochishda ofstream dan foydalaniladi. O'qish va yozish uchun faylni ochishda fstream dan foydalanasiz.

Fayl nomlari katalog yo'nalishi ma'lumotlarni o;z ichiga oladi, masalan;

```
~/homework/input.dat (UNIX)

c:\homework\input.dat (Windows)
```

Qachonki fayl nomini so'zma so'z belgilaganingizda , va nom backslash xossasini o'z ichiga oladi.

(xuddi Win dows filename dek),siz har bir backslashni ikki marta ishlatishni ta'minlashingiz kerak:

in_file.open("c:\\homework\\input.dat");

Recall that a single backslash tom ma'nodagi so'zma so'z yolg'iz backslash bayonoti bu qochish xarakterida, qaysiki boshqa xususiyat bo'lgan maxsus ma'no shakli bilan bog'langan, masalan: \n yangi satr uchun. \\ kombinatsiysi yolg'iz backslash ma'nosini anglatadi.

Agarda string variable da saqlangan nomni o'tqazishni xohlasangiz, C++ string ni C string ga o'zgartirish uchun c_str funksiyasidan foydalaning:

```
cout << "Please enter the file name:";

string filename;

cin >> filename;

ifstream in_file;

in_file.open(filename.c_str());
```

Qachonki programma tugallanganda, siz ochgan barcha stream lar avtomatik tarzda yopiladi. Bundan tashqari, qo'lda yaqin komponenti funksiyasi bilan stream ni yopish mumkin:

in_file.close();

Agar stream variable dan boshqa fayl bilan ma'lumot almashish uchun yanafoydalanishni xohlasangiz Manual yopish aynan zarur.

Fayl nomlari katalog yo'nalishi ma'lumotlarni o;z ichiga oladi, masalan;

```
~/homework/input.dat (UNIX)
```

```
c:\homework\input.dat (Windows)
```

Qachonki fayl nomini so'zma so'z belgilaganingizda , va nom backslash xossasini o'z ichiga oladi.

(xuddi Win dows filename dek),siz har bir backslashni ikki marta ishlatishni ta'minlashingiz kerak:

```
in_file.open("c:\\homework\\input.dat");
```

Recall that a single backslash tom ma'nodagi so'zma so'z yolg'iz backslash bayonoti bu qochish xarakterida, qaysiki boshqa xususiyat bo'lgan maxsus ma'no shakli bilan bog'langan, masalan: \n yangi satr uchun. \\ kombinatsiysi yolg'iz backslash ma'nosini anglatadi.

Agarda string variable da saqlangan nomni o'tqazishni xohlasangiz, C++ string ni C string ga o'zgartirish uchun c_str funksiyasidan foydalaning:

```
cout << "Please enter the file name:";
string filename;
cin >> filename;
ifstream in_file;
in_file.open(filename.c_str());
```

Qachonki programma tugallanganda, siz ochgan barcha stream lar avtomatik tarzda yopiladi. Bundan tashqari, qo'lda yaqin komponenti funksiyasi bilan stream

ni yopish mumkin:

```
in_file.close();
```

Agar stream variable dan boshqa fayl bilan ma'lumot almashish uchun yana foydalanishni xohlasangiz Manual yopish aynan zarur.

10.1.2 Fayl dan o'qish

File stream dan ma'lumot o'qish bu to'liq to'g'rilik: Siz shunchaki har doim cin dan o'qish uchun foydalangan funksiyalarni ishlatasiz:

```
string name;
double value;
in_file >> name >> value;
```

Muvaffaqiyatsizlikka uchragan funkisayalardan ma'lum bo'ladiki input ham qulagan. Siz allaqachon cin bilan bu funksiyadan foydalangansiz, konsol kiritish xatolar uchun tekshirish.

File stream larga ham shu tarzda yondashiladi. Qachonki fayldan son o'qishga harakat qilsangiz, va keyingi ma'lumotb birligi bu prop-erly formatidagi son emas, so'ngra stream omadsizlikka uchraydi. Ma'lumotni o'qigandan so'ng, qayta ishlash muvaffaqiyati uchun oldin sinash kerak:

```
if (!in_file.fail())
{
Kiritish jarayoni.
}
```

shu bilan bir qatorda, siz bir faktdan foydalanishingiz mumkin >> operator “not failed” sharoitiga aylanadi, ya'ni, input bayonoti va test sinovi imkoniyatini bergan holda:

```
if (in_file >> name >> value)
{
```

```
Kiritish jarayoni.

.

}
```

Bundan tashqari, agarda faylni ochsangiz va nom mantiqan izchil emas, yoki bu nomdagi fayl mavjud emas, keyin bu file stream ham omadsizzlikka uchragan maqomiga ega bo'ladi. Mag'lubiyatlarni kechiktirmasdan sinab ko'rish bu yaxshi fikr.

10.1.3 Faylga yozish

Faylga yozish uchun, siz ofstream yoki fstream variable ni aniqlang va uni oching, keyin output file ga ma'lumot jo'nating, bir xil operatsiyalardan foydalangan holda, qaysiki cout lar bilan:

```
ofstream out_file;

out_file.open("output.txt");

out_file << name << " " << value << endl;
```

10.1.4 Fayl ma'lumot almashishga misollar

Quyida fayldagi ma'lumotlarni almashtirishning oddiy namunasi keltirilgan. Ijtimoiy xavfsizlik boshqarmasi eng mashxur chaqaloqlar ro'yhatini o'zlarining web saytlarida e'lon qilishdi:their ularning web saytlaridan, http://www.ssa. gov/ OACT/babynames/. Agar berilgan o'n yil davomida 1000 eng mashhur nomlar so'rov bo'lsa, brauzer ekranida natija ko'rsatadi (1-rasmga qarang). Ma'lumotlarni matndek saqlash, shunchaki uni tanlang va faylga natija joylashtirish. Bu kitoblarning hamroh kodi babynames.txt deb nomlangan faylni, ya'ni 1990 yillardagi ma'lumotlarni qamragan, o'z ichiga olgan. Fayldagi har bir satr 7 ta kirisgdan tashkil topgan;

- Unvon (1 dan 1,000 gacha)
- Ism, tezlik, va erkaklar va ayollar ismlarining foizlari
- Ism, tezlik, va erkaklar va ayollar ismlaring foiz hisobi

Masalan, bu qatordagi

10 Joseph 260365 1.2681 Megan 160312 0.8168

Ko'rinadiki 260,365 ta tug'ilishdan har 10- bolaning ismi Josef bo'lgan yoki osha davrdagi barcha tug'ilishlarning 1.2681 foizi. Har 10-chi qizning eng keng tarqalgan ismi Megan bo'lgan. Nimaga bu yerda Joseflar Meganlardan ko'proq ? Ota-onalar, ularning qiz nomlari kengroq tarqalgan majmuyidan foydalanishgan ko'rinadi. Keling, ro'yxatda o'g'il-qizlar yuqori 50 foizgacha berilgan nomlarini aniqlash tomonidan, deb gumonda sinov qilaylik. Har bir chiziqni qayta ishlash uchun, biz birinchi navbatda o'qib:

```
int rank;
in_file >> rank;
```

Biz keyin bolaning ismi uch qadriyatlar majmini o'qidik

```
ism:
string name;
int count;
double percent;
in_file >> name >> count >> percent;
```

keyin biz qizlar uchun bosqichni takrorlaymiz. Chunki harakatlar o'ziga xos, Biz buning uchun yordamchi funktsiya process_name ni ta'minlaymiz. 50 foizga yetgandan so'ng jarayonni to'xtatish uchun biz tezlikni qo'shishimiz mumkin va 50 foizga yetgandan so'ng to'xtatamiz. Shunga qaramasdan, u ancha oddiyroq bo'lib chiqdi, boshlang'ich jami 50 va chastotalarni olib tashlash. Biz o'g'il bolalar va qizlar uchun alohida jamuljamlikni ajratishimiz kerak. Qachonki jami 0 dan pastga tushish pasayganda, chop etishni to'xtatamiz.

Quyidagi kod process_name funktsiyasi in-file parametr o'zgaruvchan mos yozuvlar parametr ekanligini unutmang. O'qish va yozish o'zgartiruvchilari stream variable dir. Stream variable monitorlari qancha xususiyatlar o'qilgan yoki yozilgan. har bir o'qish va yozish operatsiyalari ma'lumotlarni o'zgartiradi. Shu

sababdan, siz har doim stream parametr o'zgaruvchilar mos yozuvlar parametrlarini aniqlash kerak. To'liq programma pastda keltirilgan. Ko'rib turganingizdek, bir fayl o'qish o'qish klaviatura kiritish kabi osondir. Dastur ishlab chiqarishga qarang. Hayratlanarlisi, atigi 69 o'g'il bolalar ismi va qizlar ismining 153 tasi jami tug'ilishlarining yarmini tashkil etgan. Bu shaxsni ko'rsatuvchi biznes bilan shug'ullanuvchilar uchun yaxshi yangilik.

```
#include<iostream>
#include<string>
#include<fstream>
using namespace std;
int soni;
void fayldan_oqish(ifstream& oqi, string name[]){
    if(oqi==NULL){cout<<"Fayl topilmadi"<<endl;}
    int i=0;
    while(!oqi.eof()){
        oqi>>name[i]; soni+=1;
        cout<<name[i]<<endl; i++;
    }
}
void faylga_yozish(ofstream& yoz, string name[]){
    if(yoz==NULL){cout<<"Fayl topilmadi"<<endl;}
    for(int i=0; i<soni;i++){
        yoz<<name[i]<<endl;
    }
```

```
}
int main(){
    cout<<"Qizlar ismi"<<endl;
    string name_qizlar[100];
    ifstream oqi("qizlar_ismi.txt");
    fayldan_oqish(oqi,name_qizlar);
    ofstream yoz("qizlar_yangi.txt");
    faylga_yozish(yoz,name_qizlar);

    cout<<endl<<"Bollar ismi"<<endl;
    string name_bollar[100];
    ifstream oqi_b("bollar_ismi.txt");
    fayldan_oqish(oqi_b,name_bollar);
    ofstream yoz_b("bollar_yangi.txt");
    faylga_yozish(yoz_b,name_bollar);
}
```

10.2 Kiruvchi ma'lumotlarni o'qish

Bu bo'limda, Siz jarayonlar qanday ishlashini va haqiqiy hayotda uchraydiga ma'lumotlarni o'rgansiz.

10.2.1 So'zlarni o'qish

Siz allaqachon fayllardan >> operatori yordamida stringlarni o'qishni o'rgandingiz.

```
string word;
in_file >> word;
```

Bu yerda , bu operatsiya amalga oshiriladi , nima aniq emas . Birinchidan, oq oraliq har qanday kiritish belgilar oqimi chiqariladi, lekin ular so'zga qo'shib bo'lmaydi. Oq bo'sh joy bo'sh joy , tab belgi va yo'nalishlarini ajratib yangi satr belgi o'z ichiga oladi. oq bo'sh joy emas birinchi belgi string so'zning birinchi xarakterli - ter bo'ladi. Qo'shimcha belgilar boshqa oq bo'sh joy char - acter uchraydi , yoki fayl oxiri etib kelmoqda yo qadar qo'shiladi. so'zdan keyin oq bo'sh joy oqimi olib emas.

10.2.2 O'qish xususiyatlari

So'zma so'z ma'lumot o'qimay siz gets funksiyasi yordamida simbollarni o'qishingiz mumkin:

```
char ch;

in_file.get(ch);
```

get funksiyasi “Omadsizlikka uchramadi” statusini qaytaradi. quidagi jarayon

fayldagi barcha ma'lumotlarni o'qiydi.

```
while (in_file.get(ch))

{

Process the character ch.

}
```

get funksiyasi bo'sh joyni ham o'qiydi. Bu juda as qotadi agar siz bo'sh joy tab yoki yangi qator qidirayotgan bo'lsangiz. Boshqa tarafdan agar sizga bo'sh joy muhim bo'lmasa >> operatoridan foydalanganiz ma'qul.

```
in_file >> ch; // ch is set to the next non-white space character
```

Agar belgi o'qish va uni pushaymon bo'lsangiz, uni unget mumkin, shunday qilib , keyingi kiritish operatsiya yana o'qish mumkin . Ammo, faqat oxirgi belgini unget mumkin. Bu bir - belgi lookahead deyiladi. Siz kiritish oqimi keyingi xarakteriga qarash bir imkoniyat , va siz uni iste'mol yoki orqa qo'yish xohlaysizmi

Agar qaror qabul qilish mumkin.

```
rqam qidirish kodi:
char ch;
in_file.get(ch);
if (isdigit(ch))
{
in_file.unget();
int n;
data >> n; // raqamni o'qish ch bilan boshlanadi.
}
```

10.2.3 Liniyalarni o'qish

Har bir qator kerakli ma'lumotdan iborat bo'ladigan bo'lsa getline funksiyasidan foydalanilgan ma'qul.

```
string line;
getline(in_file, line);
while (getline(in_file, line))
{
Jarayon line bo'yicha.
}
```

Jadval 1 <cctype>dagi funksiya xususiyati

Funksiy	Qabul qilingan tip
isdigit	0 ... 9
isalpha	a ... z, A ... Z
islower	a ... z
isupper	A ... Z
isalnum	a ... z, A ... Z, 0 ... 9
isspace	Bo'sh joy(space, tab, newline, and the rarely used carriage return, form feed, and vertical tab)

Bu yerda fayl ichidagi satrlar jarayoniga misollar ko'rsatilgan. Odamlar soni haqida ma'lumot faylini joylashgan CIA World Factbook saytidan (https://www.cia.gov/library/publications/the-world-factbook/index.html) quyidagicha satrlar olingan:

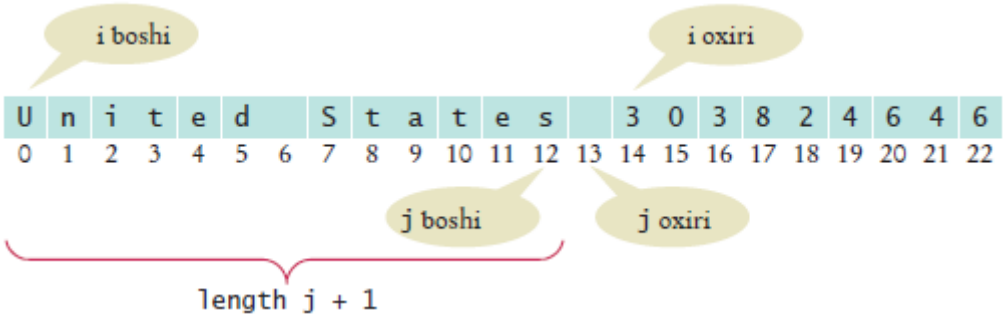
```
Xitoy 1330044605

Hindiston 1147995898

AQSH 303824646

...
```

Sabab har qanday satrlar ma'lumoti yozib olingan, bu string tipiga satr o'zgaruvchini o'girib olish uchun getline funksiyasidan foydalaniladi. String tipidan chiqarib olinadi. Birinchi raqam joylashgan joy:

```
int i = 0;

while (!isdigit(line[i])) { i++; }
```

So'ng teskarisiga boradi va bo'sh joyni tashlab ketadi:

```
int j = i - 1;

while (isspace(line[j])) { j--; }
```



Yakunda, davlat nomi va odamlari soni ko'chirib olindi:

```
string country_name = line.substr(0, j + 1);

string population = line.substr(i);
```

Aniq bir muammo bor. Odamlar soni string tipida saqlanadi raqam

ko`rinishida emas. Siz ko'rib o'tgan 8.4 seksiyada odamlar soni raqam tipidan foydalanib yozilgan.

10.3 Chiqaruvchi matnlarni yozish

Oqim satrlari va raqamlarni >> operatori orqali yuborasiz. Oqimning biror belgisini yozish uchun, foydalangani.

```
out_file.put(ch);
```

Chiqish formatlashtirilganda nazorat qilish uchun siz manipulatoridan foydalaning. Manipulator oqimning ta'sir qiymati hisoblanadi. Bu << operatori yordamida bir oqim yuboriladi. Setw manipulatorni siz allaqachon ishlatganingizga misol.

```
Namuna:
out_file << setw(10);
```

Chiqarish har doim bevosita sabab bo'lmaydi, lekin keyingilari yozib bo'linganda, qoldirilgan yetarlicha interval 10 ta xarakterga ega. (Agar kiritilgan qiymat maydonga mos kelmasa, bu ko'chirib olinmaydi.) Vaqti vaqti bilan nollarni kiritishingizga to'g'ri keladi, masalan soat va minutni 09:01 ni chop etish uchin. Bu bilan manipulator to'ldiriladi:

```
out_file << setfill('0') << setw(2) << hours << ":"
<< setw(2) << minutes << setfill(' ');
```

Hozir 0 bo'shliqni to'ldirishda ishlatiladi. Keyin xarakterlar qayta saqlanadi. Xatolar orqali, to'ldirilgan xarakterlar punktdan oldin paydo bo'ladi:

```
out_file << setw(10) << 123 << endl << setw(10) <<
4567;
natija
123
4567
```

Raqamlar qatori chapga o'tkaziladi. Bu tizim ishlashi raqamlar uchun yaxshi, lekin string tipidagilar uchun emas. Odatda, siz belgilarni o'ngdan qatorga o'tkazishni xoxlasangiz. Tizimni tanlash uchun o'ng va chap manipulyatorlardan foydalanasiz.

```
out_file << left << setw(10) << word << right << setw(10) << number;
```

Majburiyatni bajarmaydigan formatdagi butun sonlar general sonlar deyiladi. Bu format raqamlarni aniqlik bilan ko'rsatadi, ilmiy ishoralarni katta va kichik sonlar bilan ko'rsatadi.

```
out_file << 12.3456789 << " " << 123456789.0;
```

```
natijada
```

```
12.3457 1.23457e+08
```

Belgilangan formatda barcha qiymatni shu sonni raqamidan keyingi o'nlik nuqta bilan chop etadi. ushbu formatda, bir xil sonlar quyida ko'rsatilgan. 12.345679 123456789.000000

Belgilangan manipulyatoridan foydalanish ushbu formatni tanlab, va aniq joylashtirish manipulyatoridan aniq o'zgartirish uchun:

Misol uchun,

```
out_file << fixed << setprecision(2) << 1.2 << " " << 1.235
```

```
natijad a 1.20 1.24
```

Stream Boshqaruvchilari			
Boshqaruv	Maqsad	Misol	Chiqish
setw	keyingisini joylashtirish	out_file << setw(6) << 123 << endl << 123 << endl << setw(6) << 12345678;	123 123 12345678
setfill	Bosh joyga to'ldirilgan xarakteristikalarni joylashtirish	out_file << setfill('0') << setw(6) << 123;	000123
left	Chap tizimni belgilash.	out_file << left << setw(6) << 123;	
123 right	tizimni belgilash.	out_file << right << setw(6) << 123;	123
fixed	Butun-nuqtali sonni belgilash-.	double x = 123.4567; out_file << x << endl << fixed << x;	123.457 123.456700
setprecision	Belgilangan sonlar, raqamardan keyin o'nli nuqta belgilangan formatda berilgan.	double x = 123.4567; out_file << fixed << x << endl << setprecision(2) << x;	123.456700 123.46

Yakunda ikkita oqim manipulyatori bor. Shu barcha manipulyatorlar hamma songi jarayonlar uchun oqim obyektlarning qismi joylab setw bilan yoziladi . Har bir natija jarayonida bosh joylarga 0 qayta joylanadi. Bu manipulyatordan foydalanishda `<iomanip>` asosiy kutubxana hisoblanadi.

10.4 Satr oqimlari

O'tgan qismda, siz faylda oqim fayllarini o'qish harakteristikalarini va faylda oqim fayliga yozish xarakteristikalarini ko'rgansiz. `istringstream` sinfi satrni o'qiydilar va `ostringstream` satrni yozadilar . Bu ovoz chiqarmaydi shunday ajoyib— biz oldindan bilamiz ma'lumotlarni va o'zgartirilgan satr xarakteristikalarini. Biroq, oqim satri sinflarida bir xil interfacelar bor boshqa oqim sinflari kabi. Ba'zi xollarda, siz tanish >> va << operatorlaridan satrda sonlarni o'qish va yozish uchun foydalanasiz. `istringstream va ostringstream` sinflari *moslashtiruvchilar deb ataladi* —bu satr adapterlari oqim interfeysi uchundir. Bu `<sstream>` kutubxona asosiy satrga oid kutubxona . Bular turli misollar. O'ylangan satr sanalar quyidagicha "January 24, 1973" va biz ularni oy, kun va yilga ajratishimiz kerak . Birinchi, `istringstream` obyektini tuzamiz. Bunda str funksiyasidan foydalanamiz siz buni o'qishingiz uchun:

```
istringstream strm;
strm.str("January 24, 1973");
```

Keyin, faqat >> operatoridan foydalanib oy nomini , kun, alohida vergul bilan, and yilni o'qish:

```
string month;
int day;
string comma;
int year;
strm >> month >> day >> comma >> year;
```

Hozir oy "January", kun 24, va yil 1973. Kiritishda natijalar kun va yil butun sonlarda yoziladi. Alohida substrni olamiz, biz faqat belgilar olganmiz, sonlar emas. Natijadad, o'gartirlayotgan belgilar bu sonlarni o'z ichiga oladi, ular shunday umumiy natijaga yordam beruvchifunksiyali butun qiymatlar:

```
int string_to_int(string s)
{
istringstream strm;
strm.str(s);
int n = 0;
strm >> n;
return n;}
```

Misol uchun, string_to_int("1973") butun qiymat 1973. String tipidagilarni yozish uchun, siz string tipida integer yoki float tipiga o'zgartirishingiz lozim.

```
ostringstream object:
ostringstream strm;
```

Keyin, << operatoridan foydalanib qatorga sonlarni yozamiz. Raqamlar

ketmaketlik bilan o'zgartirilgan :

```
strm << fixed << setprecision(5) << 10.0 / 3;
```

Hozir qator belgilarni o'z ichiga olgan "3.33333".Bu belgilarni qatordan olishda, str funksiyasi chaqiriladi:

```
string output = strm.str();
```

Siz ko'p satrlar kompleksini qurishda yo'lida. Bu yerda biz satrli ma'lumotlarni quramiz oy, kun va yillarni kiritib:

```
string month = "January";

int day = 24;

int year = 1973;

ostringstream strm;

strm << month << " " << day << ", " << year;

string output = strm.str();
```

Hozir "January 24, 1973" satrli natija chiqdi. Biz integerdagi kun va yilni satrdan o'zgartirib yozamiz . Yana, ostringstream strm integerni stringga o'zgartirishda umumiy operatsiyalarni o'z ichiga oluvchi natija uchun foydali funksiylardir:

```
string int_to_string(int n)

{

ostringstream strm;

strm << n;

return strm.str();

}
```

Misol uchu, int_to_string(1973) satrlar- "1973".

10.5 Buyruqlar qatori argumentlari

Operatsion sistemaga bog'liq bo'lgan va C++ ning so'ngi o'zgarishlardan foydalanilgan, bu yerda dastur turli xil metodlar bilan boshlangan—misol uchun, kompilatsiya jarayonida “Run”ni belgilash orqali, dastur ikonkasini bosib yoki tez chiquvhi windows qobig'i buyruqlariga dastur nomini yozish orqali. Xarflar metodi “buyruqlar qatoridan dasturni chaqirish” deb ataladi. Qachonki siz bu metoddan foydalanganingizda, dastur nomini yozishga albatta majbursiz , lekin siz bu dasturdan foydalananishda qo'shimcha ma'lumotlarni yana yozasiz. Bunda qo'shimcha satrlar buyruqlar qatori argumentlari deb ataladi. Misol uchun, agar siz dasturni buyruqlar qatori bilan ishga tushirsangiz `prog -v input.dat` bu dastur ikkita buyruqlar qatori argumentlarini qabul qiladi: `satrlar "-v" va "input.dat".` Ular odatda satrlar boshlanishida chiziqcha va boshqa satrlar fayli nomi bilan tarjima qilinadi. Buyruqlar qatori argumentlarini qabul qilishda, turli yo'llarda asossiy funksiyani aniqlab olishingiz kerak. Siz ikkita o'zgaruvchi parametrni aniqlab olishingiz kerak: butun va char* tipidagi asosiy belgilarining tartibini.

```
main(int argc, char* argv[])
{
...
}
```

Bu yerda argc argumentlarni sanaydi va argv argumentlar qiymatini o'z ichiga oladi. Bizning misolimizda, argc 3ga teng va argv uch satrni oz ichiga oladi

```
argv[0]: "prog"
argv[1]: "-v"
```

`argv[2]: "input.dat" buni yozamiz argv[0] dastur nomida argc bo'lib keladi.`

10.6 Ixtiyoriy bog'lanish va ikkilik fayllar

Keyingi qismlarda siz ma'lumotlarni faylning ixtiyoriy joyiga yozish xamda o'qishni va rasmli ma'lumotlarni o'zgartirishni o'rganasiz.

10.6.1 Ixtiyoriy ulanish

Hozircha, siz oldinga yoki arqaga o'tmasdan turib fayldan bir vaqtning o'zida ma'lumotni o'qidingiz va yozdingiz. Bunday bog'lanish izchil (ketma-ket) bog'lanish deyiladi. Ko'pgina murojaatlarimizda fayldagi muayyan ma'lumotga undan avvalgilarini o'qib o'tirmasdan murojaat qilishni hohlaymiz. Bu bog'lanish turi tasodifiy bog'lanish deb ataladi (3-shakldan ko'ring). Tasodifiy bog'lanishda hech narsa "tasodifiy" bo'lmaydi—bu atama shuni anglatadiki siz faylning har qanday qismidagi har qanday ma'lumotni o'qishingiz yoki ifodalashingiz mumkin. Faqatgina fayl oqimlari tasodifiy bog'lanishga imkon beradi; terminal va klaviaturaga bog'langan cin (kiritish) va cout (chiqarish) oqimlaridan tashqari. Bu fayl oqimlari alohida mavqelarga ega: qo'yish (put) mavqei va olish (get) mavqei 4- shakldan ko'ring). Bu holatlar keyingi belgi qayerga yozilgan yoki o'qilganligini ko'rsatadi. Quyidagi funksiya berilgan qiymatga get (olish) va put (qo'yish) amallarini chaqiradi va oqim boshidan hisoblanadi.

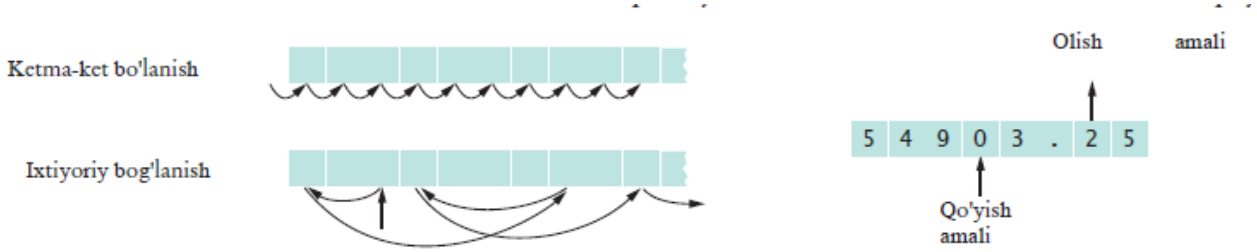
```
strm.seekg(position);

strm.seekp(position);
```

Galdagi qo'yiladigan yoki olinadigan qiymatni hisoblash (fayl boshidan boshlab hisobga olinadi) uchun quyidagidan foydalaniladi

```
position=strm.tellg();

position=strm.tellp();
```

10.6.2 Ikkilik fayllar

Ko'pgina fayllar, hususan rasm va ovozlarni o'zida jamlagan fayllar, ma'lumotni tekst ko'rinishida saqlamaydi, balki ikkilik kodlar ko'rinishida saqlaydi. Raqamlar



izchillik bilan **bayt**larda ifodalani, huddi bu kompyuter hotirasidadagi kabi. (Har bir bayt 0 dan 255 oraliqda baholanadi.) Ikkilik formatda, haqiqiy raqamlar 8 baytni egallaydi. Biz rasmli ikkilik formatdagi fayllar uchun ixtiyoriy bog'lanishni o'rganamiz. Biz ikkilik fayllar haqida bir oz texnik ma'lumotga ega bo'lishimiz kerak. Ikkilik faylni ochib yozish yoki o'qish uchun quyidagi komandadan foydalaniladi:

```
fstream strm;
strm.open(filename, ios::in | ios::out |
ios::binary);
```

Bir baytni o'qish uchun so'rov

```
int input = strm.get();
```

Bu so'rov 0 va 255 orasidagi qiymatni qaytaradi. Raqamni o'qish uchun, to'rt baytni o'qish uchun b_0, b_1, b_2, b_3 va ularni qo'shish $b_0 + b_1 \_ 256 + b_2 \_ 256_2 + b_3 \_ 256_3$. Biz bu topshiriqqa yordamchi funksiyani yuboramiz. Ikkilik fayldan raqamlarni o'qishda operatordan foydalanib bo'lmaydi.

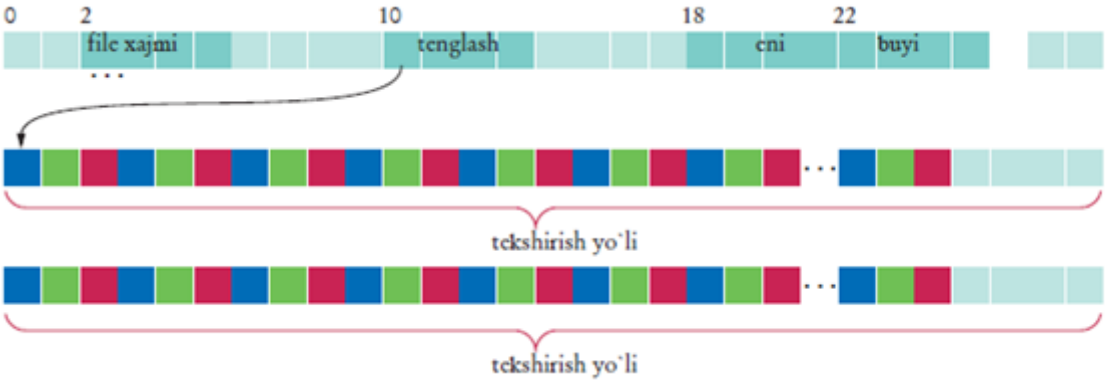
10.6.3 Rasmli fayldagi jarayonlar

Bu qismda BMP formatli rasmli faylga o'zgartirish kirituvchi programmaga qanday kod yozishni o'rganasiz. BMP odatiy GIF, PNG, va JPEG formatli fayllarga o'xshamagan, chunki u siqilgan ma'lumotlardan foydalanmaydi. Shunga ko'ra BMP fayllar juda katta bo'lib siz uni kamdan kam hollarda ushratasiz. Biror rasmli faylni BMP formatga qanday o'girsa bo'ladi. BMP formatning turli hil versiyalari bor; biz ko'pincha 24-rangli formatli oddiy va sodda turini tanlab olamiz. Bu formatda, har bir piksel uch baytli ketma ketlikni tashkil qiladi , ular ko'k, yashil, va qizil qiymatlar. Masalan, ko'k rang (ko'k va yashil ranglar

aralashmasi) bu 255 255 0, qizil esa 0 0 255, va kulrang 128 128 128.

BMP fiayl turli ma'lumot qismlaridan tashkil topgan bosh qismdan boshlanadi. Bizga faqatgina quyidagilar kerak:

O'rni	Bo'lim
2	Fayl bayt hajmi
10	Rasim boshlanishi
18	Rasim piksellari eniga
22	Rasim piksellari bo'yiga



Rasm quyi qator piksellaridan boshlab pikselli qatorga ketma ketlikda saqlanadi. Har bir piksel qatori ko'k/yashil/qizil uchlikni o'z ichiga oladi. Qator ohiri qo'shimcha baytlar bilan to'ldiriladi shunday qilib qatordagi baytlar soni 4 ga karrali bo'ladi. (5-rasm.) Masalan, agar qator shunday uch pikselli,biri ko'k, biri qizil va kulrangdan iborat bo'lsa qator quyidagicha kodlanadi 255 255 0 0 0 255 128 128 128 x y z buyerda $x\,y\,z$ lar bilan qator to'ldirilib 12 ga tenglanmoqda, bu esa 4 ga karrali bo'ladi. Bu real fayl formatlari bilan ishlashga undovchi qiziqarli tajriba ishiga o'xshaydi. Bo'linma ohiridagi na'muna dasturi BMP fayldagi har bir pikselni o'qiydi hamda ularni qarama qarshi rangi bilan almashtiradi, oqni qoraga, ko'kni qizilga va h.k. Natijada suratga olishda foydalanilgan qadimgi film kameralari olinganidek suratning teskari ranglisi hosil bo'ladi.

Bu dasturni ishlatib ko`rish, bitta yoqtirgan rasmingizni oling,BMP formatga o`tkazish uchun foydalaning (yoki bu kitob kodlaridagi shu fayldan queen-mary.bmp), keyin ishga tushirib natijani ko`ring P8.21 va P8.22 mashqlarda ko`p qiziqarli faktlar so`ralgan.

1-misol.

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main(){
  FILE *fin, *fout;
  fin=fopen("oqi.dat","rb");
  if(fin==NULL){cout<<"fayl topilmadi"<<endl;
   exit(1);   }
```

```
    int n;  fscanf(fin,"%d",&n);
  fout=fopen("yoz.dat","wb");
  fprintf(fout,"n= %d",n);
  cout<<"n= "<<n<<endl;  fclose(fin);fclose(fout);}
```

2-misol.

```
#include<iostream>
#include <stdio.h>
using namespace std;
int main(){
const int N = 10;
int B[N],A[N]={11,2,3,4,51,6,7,81,91,10},i=0, n;
FILE *fp, *fp1; int j=0;
 fp=fopen("input_2.dat","w");
 for(int i=0;i<N;i++)
 fprintf(fp,"%d \n",A[i]); }
```

3-misol.

```
#include<iostream>
using namespace std;
int main(){
  FILE *fin; int N=3;
  fin=fopen("yoz.dat","rb");
  int A[100];
 if(fin == NULL ){
  cout<<"fayl ochilmadi.";
```

```
  exit(1);  }
 int n=fread(A,sizeof(int),10,fin);
  if ( n < N )
cout<<"faylda ma’lumotlar yetishmayapdi";
  fclose(fin);
  for(int i=0;i<10;i++){
    cout<<A[i]<<" ";  }}
```

4-misol.

```
#include<iostream>
using namespace std;
int main(){
  FILE *fin;
  fin=fopen("yoz.dat","wb");
  int A[100]={1,2,3,4,5,6,7,8,9,10};
  fwrite(A,sizeof(int),10, fin); fclose(fin);}}
```

Mavzu doir test savollari:

6. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
 FILE *fin, *fout;
 fin=fopen("oqi.dat","rb");
 if(fin==NULL){cout<<"fayl topilmadi"<<endl;
  exit(1);  }
  else{ cout<<"Fay topildi"<<endl;
    }
}
```

e) Fayl topilmadi
f) Xatolik
g) Gramatik xatolik
h) Fayl topildi

7. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
 FILE *fin, *fout;
 fin=fopen("oqi.dat","rb");
```

```
 if(fin==NULL){cout<<"fayl topilmadi"<<endl;
  exit(1);  }
   else{ cout<<"Fay topildi"<<endl;
     }
     int n;
   fscanf(fin,"%d",&n);
   cout<<n<<endl;
}
```

a) Fayl topilmadi 1
b) Xatolik 9
c) Gramatik xatolik
d) Fayl topildi 0

8. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
 FILE *fin, *fout;
 fin=fopen("oqi.dat","rb");
 if(fin==NULL){cout<<"fayl topilmadi"<<endl;
  exit(1);  }
   else{ cout<<"Fay topildi"<<endl;
     }
     int n;
   fscanf(fin,"%d",&n);
   fout=fopen("yoz.dat","wb");
   fprintf(fout,"n= %d",n);
   cout<<n<<endl;
}
```

a) Fayl topilmadi, faylga yozildi
b) Xatolik 1
c) Gramatik xatolik
d) Fayl topildi 0 ni faylga yozadi

9. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
int main(){
int legth;
char *buffer, fayl[]="matn.txt", yangi[]="y_fayl.txt";
ifstream fayldan(fayl, ios::binary);
if(!fayldan.is_open()){
     cout<<"Fayl ochilishda hatolik yuz berdi";
     exit(1);        }
ofstream faylga(yangi, ios::binary);
fayldan.seekg(0,ios::end);
legth = fayldan.tellg();
```

```
fayldan.seekg(0,ios::beg);
buffer =  new char[legth];
fayldan.read(buffer, legth);
fayldan.close();
faylga.write(buffer, legth);
delete[] buffer;
cout<<"Fayl nushalandi"<<endl;  }}
```

a) Fayl nushalandi
b) Fayl ochilishda hatolik yuz berdi
c) Gramatik xatolik
d) Fayl topildi

10. Quyidagi ifoda qanday natija chiqaradi?

```
int main(){
const int N = 10;
int B[N],A[N]={11,2,3,4,51,6,7,81,91,10},i=0, n;
FILE *fp, *fp1; int j=0;
 fp=fopen("input_2.dat","w");
 for(int i=0;i<N;i++)
 fprintf(fp,"%d \n",A[i]);
}
```

a) Fayl topilmadi
b) Fayl ochilishda hatolik yuz berdi
c) Gramatik xatolik 0 chiqadi
d) Xech nima chiqaydi

Nazorat savollari:

1. Fayllar bilan ishlashning qandey usullari bor?
2. Matnli faylga yozish uchun qaydey funksiya ishlatiladi?
3. Matnli fayldan o’qish uchun qaydey funksiya ishlatiladi?
4. Binar faylga yozish uchun qaydey funksiya ishlatiladi?
5. Binar fayldan o’qish uchun qaydey funksiya ishlatiladi?
6. Binar fayllardan nima maqsadda foydalniladi?
7. Matnli fayl bilan binary faylni qandey farqli tomonlari bor?
8. Binar fayllar xotiradan joy egallaeydimi? Agar egallasa qancha?
9. fprintf funksiyasining vazifasi nimalardan iborat?
10. Scanf funksiyasining vazifasi nimalardan iborat?
11. ifstreamning vazifasi nimalardan iborat?
12. ofstreamning vazifasi nimalardan iborat?

11. Ob’ektga yo’naltirilgan dasturlash asoslari

Bu bobda siz obyektga yo’naltirilgan dasturlash bilan tanishasiz, bu murakkab dasturlarni yozishda muxim uslub hisoblanadi. Obyektga yo’naltirilgan dasturlashda siz nafaqat raqam va qatorlarni boshqara olasiz, balki dasturiy vositangiz uchun kerakli bo’lgan obyektlar bilan ishlaysiz. Bir xil harakatdagi obyektlar (chap tarafga aylanuvchi tegirmon) sinflarga guruh sifatida ajratilgan. Dasturchi istalgan harakatni ushbu sinflarga funksiyalarni aniqlash va joriy qilish orqali ta’minlaydi. Bu bobda, siz o’z sinfingizni qanday qilib topishni, aniqlashni va joriy qilishni o’rganasiz va bularni o’z dasturingizda ishlata olasiz.

11.1 Obyektga yo’naltirilgan dasturlash

Dasturlaringizni qanday tuzishni, vazifalarni funksiyalarga ajratish orqali o’rgandingiz. Bu juda soz. Dastur katta bo’lishi bilan funksiyalarning ulkan to’plamini boshqarish tobora qiyinlashadi. Bu muammoni yechish uchun kopmpyuter olimlari obyektga yo’naltirilgan dasturlashni kashf qildilar. Har bir obyekt o’z ma’lumotlar to’plamiga ega bo’lib, funksiyalar to’plami bilan birgalikda ma’lumot ustida ishlay oladi. (Bu funksiyalar komponentlik funksiyalari deb ataladi). Siz allaqachon obyektga yo’naltirilgan dasturlashni tajribangizda sinab ko’rdingiz, bu xol string obyektlari yoki cin va cout oqimlarini ishlatganingizda yuz bergan. Masalan, siz length va substr komponentlik funksiyalarini string obyektlari bilan ishlaganda ishlatgansiz.Oqimlar uchun ishlatiladigan >> va << operatorlari ham komponentlik funksiyalari kabi joriy etiladi. — 406 betdagi 9.2 maxsus mavzuga qarang. C++ da, dasturchi yagona obyektni joriy qilmaydi. Uning o’rniga, dasturchi sinf bilan ta’minlaydi. Sinf birxil harakatdagi obyektlar to’plamini tasvirlaydi. Masalan, string sinfi barcha qatorlar harakatini tacvirlaydi. Sinf qator elementlarni qandaysaqlashini, qaysi komponentlik funksiyalarini qatorda ishlatishni va komponentlik funksiyalarini qanday joriy qilishni aniqlaydi.

Obyektga yo’naltirilgan dasturlashni rivojlantirganingizda, siz o’z sinflaringizni yaratasiz, ular dasturiy vositangiz uchun nima muximligini

tasvirlaydi. Masalan, talaba ma’lumotlar bazasida, siz Student va Course sinflari bilan ishlashingiz mumkin. Albatta, bu sinflar uchun komponentlik funksiyalarini qo’llashingiz kerak. Obyekt bilan ishlaganingizda, siz uning qanday joriy qilinishini bilmaysiz. Sizga string elementlar ketma ketligini qanday tashkil etishini bilish shart emas yoki cin obyekti kiruvchi ma’lumotlarni konsoldan qanday o’qiy olishi ham muxim emas.Siz bilishingi kerak bo’lgan jixat bu ochiq interfeysdir: siz faollashtira oladigan komponentlik funksiyalari xususiyatlari. Inkapsulyasiyani o’z sinflaringiz uchun qo’llamoqchi bo’lasiz. Siz sinfni tanlaganingizda, siz ochiq interfeys uchun harakatni aniqlaysiz, lekin joriy qilish detallarini yashirasiz. Inkapsulyasiya sizning sinflaringizni ishlatadigan dasturchilar uchun foyda keltiradi. Ular sizning dasturlaringizni, ularning qanday joriy qilinganligini bilmay turib, ishlata oladilar, xuddi siz string va oqim sinflarining ichki detallarini bilmay turib ishlatganiniz kabi. Inkapsulyasiya sinfning joriy etuvchisi uchun ham qulaydir. Uzoq vaqt rivojlantirib kelingan dastur ishlatilganda, joriy qilinish detallarining o’zgarishi tabiiydir, odatda bu xolat obyektlarni yanada samarali va qobiliyatli qiladi. Inkapsulyasiya bu o’zgarishlarni amalga oshirishda muhim hisoblanadi. Joriy etilish yashirin bo’lganda, joriy etuvchi uchun o’zgarishlarni amalga oshirishga to’siq yo’q. Chunki joriy eilish yashirin bo’lib, tbu o’zgarishlar obyektlarni ishlatuvchi dastrchilarga ta’sir o’kazmaydi.

Bu bobda, siz C++ o’z sinfingizni qanday yaratishni va joriy o’rganasiz, hamda o’z dasturingizni obyektga yo’naltirilgan usulda tuzishni, inkapsulyasiya ishlatilish tamoyilini bilib olasiz.

11.2 Sinf ochiq interfeysini aniqlash

Sinfni aniqlash uchun, birinchi galda biz ochiq interfeysni aniqlab olamiz. Sinf barcha komponentlik funksiyalaridan tashkil topgan bo’lib, sinf foydalanuvchisi uni o’z sinflariga joriy etishni istaydi. Kelin oddiy misolni ko’rib chiqaylik.Biz registrlarni xosil qiluv chi registrlardan foydalanamiz. Savdoni boshlamoqchi bo’lgan g’aznachi to’lovni o’tkazish uchun tugmani bosadi va har

bir xaridni o’tkaza boshlaydi. item. Monitor olingan xaridni va uning umumiy qiymatini ko’rsatib beradi. Bunda biz quyidagi operasiyalarni amalga oshirmoqchimiz:

• Xarid narxini qo’shing.

• Xaridning umumiy miqdorini va qiymatni hisoblang.

• Yangi savdoni boshlash uchun kassa apparatini tozalang.

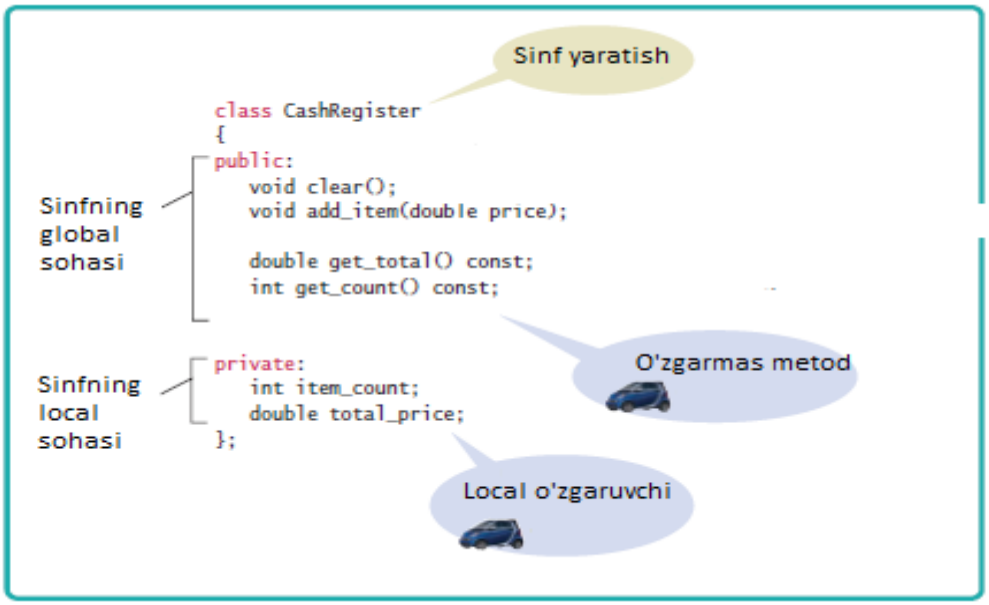
Sinfimizni `CashRegister.` deb nomlaymiz (Biz nomlash kelishuviga amal qilamiz. Nomlash kelishuvi *camel case* (tuya xolati) deb ataladi chunki xarflar orasidagi tepaga chiqib qoluvchi xarflar tuyaning o’rkachiga o’xshaydi)

Bu yerdae C++ sintaksisi uchun CashRegister sinf tavsifi keltirilgan:

```
class CashRegister
{
public:
void clear();
void add_item(double price);
double get_total() const;
int get_count() const;
private:
data members—see Section 9.3
};
```

Komponentlik funksiyalari sinfning ochiq bo’limida elon qilinadi. Dasturning har qanday qismi komponentlik funksiyasini chaqira oladi. Ma’lumot elementlari sinfning yopiq bo’limida beriladi. Faqatgina sinfning komponentlik funksiyalarigina ushbu ma’lumotlar elementidan foydalana oladi; ular dastur eslatuvchisi uchun yashirin bo’ladi. Yopiq elementlarni ochiq bo’limga yetkazishdan avval e’lon qilish mumkin, lekin bu kitobda, biz ochiq bo’limnm birinchi joylashtirganmiz. Shunday bo’lsada, sinfni o’quvchi ko’p dasturchilar

foydalanuvchilardir, joriy etuvchilar emas, va ular ko'proq yopiq joriy etilishdan ko'ra ochiq interfeysga ko'proq qiziqadilar. Komponentlik funksiyalari e'lonlari odatiy funksiya e'lonlariga o'xshaydilar. Bu e'lonlar hyech narsani joriy etmaydidar. Siz komponentlik funksiyalarini qanday joriy qilishni 9.4 bo'limda ko'rib chiqasiz. Ikki turdagi komponentlik funksiyalari mavjud: mutatorlar va aksessorlar. Mutator funksiyasi obyekt ma'lumotlar elementini yaxshilaydi.



Mutatorlar ikki xil: clear va add_item bo'ladi. Bu funksiyalarning birini chaqirsangiz, umumiy miqdor va qiymat o'zgaradi. Aksessorlar obyektni o'zgartirmay turib, biror ma'lumot uchun so'rov beradi. CashRegister cinfi ikiita aksessorga ega: get_total va get_count. CashRegister obyekti uchun bu funksiyalardan birini qo'llash, obyektni o'zgartirmay turib qiymatni osonlikcha qaytarib beradi. C++da, const zaxiradagi so'z aksessor funksiyasini belgilash uchun ishlatiladi. (402 betdagi 9.2 dasturlash maslaxatiga qarang), bu kabi:

```
double get_total() const;
```

Komponentlik funksiyalari nuqta notasiyasini ishlatish orqali faollashtirilgan bo'lib, ularni siz allaqachon qator va oqim fuyenksiyalarida ko'rgansiz:

```
CashRegister register1;
register1.clear();
```

Endi biz CashRegister obyekti nima qila olishini bilamiz, lekin qanday qilganligini bilmaymiz. Albatta, dasturlarimizda CashRegister obyektlarini ishlatamiz, bilishimiz kerak emas. Osonlikcha ochiq interfeysdan foydalanamiz.

Mutator funksiyalari massivlarda ko'rsatilgan bo'lib, ko'rsatkichlar ma'lumotni yaxshilashga undamoqda. Aksessor funksiyalari massivlarda ko'rsatilgan bo'lib, bu ma'lumot o'qishning boshqa yo'lidir.

11.3 Ma'lumotlar elementi

Obyekt o'z ma'lumotlarini ma'lumotlar elementida saqlaydi. Bular sinfni ichida e'lon qilinadigan o'zgaruvchilardir. Sinfni joriy qilganda, siz qaysi obyekt qanday ma'lumotni saqlashi kerakligini aniqlashingiz zarur. Obyekt komponentlik funksiyasini chaqirish uchun zarur bo'lgan barcha ma'lumotga ega bo'lishi kerak. Barcha komponentlik funksiyalarini ko'rib chiqib, qanday ma'lumot talablari kerakligini o'ylab ko'ring. Qabul qiluvchi funksiyadan boshlash yaxshi fikrdir. Masalan, CashRegister obyekti et_total funksiyasi uchun to'g'ri qiymatni qaytarib bera olishi kerak. Bu degani, chaqiruv funksiyasida barcha kiruvi narxlarni saqlash va umumiy natijani hisoblash osonroq bo'lishi zarur. Endi get_ count funksiyasini olish uchun kerakli ishni amalga oshiring. Agar kassa apparati barcha kiruvchi narxlarni saqlay olsa, u ularni get_ count funksiyasida sanay oladi. Aksincha, hisoblash uchun o'zgaruvchiga ega bo'lishingiz kerak.

Obyekt o'z ma'lumotlarini ma'lumotlar elementida saqlaydi. Bular sinfni ichida e'lon qilinadigan o'zgaruvchilardir. Sinfni joriy qilganda, siz qaysi obyekt qanday ma'lumotni saqlashi kerakligini aniqlashingiz zarur. Obyekt komponentlik funksiyasini chaqirish uchun zarur bo'lgan barcha ma'lumotga ega bo'lishi kerak. Barcha komponentlik funksiyalarini ko'rib chiqib, qanday ma'lumot talablari kerakligini o'ylab ko'ring. Qabul qiluvchi funksiyadan boshlash yaxshi fikrdir. Masalan, CashRegister obyekti et_total funksiyasi uchun to'g'ri qiymatni qaytarib bera olishi kerak. Bu degani, chaqiruv funksiyasida barcha kiruvi narxlarni saqlash va umumiy natijani hisoblash osonroq bo'lishi zarur. Endi get_ count funksiyasini olish uchun kerakli ishni amalga oshiring. Agar kassa apparati barcha kiruvchi

narxlarni saqlay olsa, u ularni get_ count funksiyasida sanay oladi. Aksincha, hisoblash uchun o'zgaruvchiga ega bo'lishingiz kerak.

11.4 Konstruktorlar

Konstruktor komponentlik funksiyasi bo'lib, obyekt ma'lumotlar elementini dastlabki xolatga qaytaradi. Konstruktor obyekt yaratilgan har qanday vaqt chaqiriladi. Konstruktorni kiritganda, ma'lumotlar elementining barchasi to'g'ri joylashtirilganligiga ishonch xosil qilingki, komplnentlik funksiyalari obyekt sifatida ishlamasin. Konstruktorlarning ahamiyatini tushunish uchun, quyidagilarni ko'rib chiqing:

```
CashRegister register1;

register1.add_item(1.95);

int count = register1.get_count();
```

Bu yerda dasturchi clear funksiyasini elementlarni qo'shishdan oldin chaqirishni unutgan. Shuning uchun, register1 obyekti ma'lumotlar elementi tasodifiy qiymatlar bilan dastlabki xolatga qaytarilgan. Konstruktorlar obyekt aniqlanganda to'laligicha dastlabki holatga qaytarilishiga kafolat beradi. Konstruktor nomi sinf nomi bilan bir xil bo'ladi. Siz konstruktorlarni sinf tafsifida e'lon qilasiz. Masalan:

```
class CashRegister

{

public:

CashRegister(); // A constructori

...

};
```

Konstruktor qiymatni hyech qachon qaytarmaydi, lekin siz zaxiradagi void so'zini e'lon qilinganda ishlatmaydi. Bu yerda konstruktor tavsifi berilgan:

```
CashRegister::CashRegister()
{
item_count = 0;
total_price = 0;
}
```

Konstruktor tafsifida, birinchi CashRegister (::dan oldin) biz CashRegister sinfida komponentlik funksiyasini aniqlamoqchi ekanligimizni ko'rsatadi. Ikkinchi CashRegister komponentlik funksiyasi nomidir.

Ko'rib turgan konstruktoringizning argumentlari yo'q. Bunday konstruktorlar defolt konstruktorlar deb ataladi.Ular obyektni aniqlaganingizda va konstruksiya uchun parametrlarni belgilamaganingizda ishlatiladi.

```
CashRegister register1;
```

keyin esa defolt konstruktor chaqiriladi. U register1.item_count va register1.total_ ni nolga o'rnatadi price. Ko'p sinflar have more than one constructor.

```
class BankAccount
{
public:
BankAccount();
BankAccount(double initial_balance);
private:
double balance;
};
```

11.5 Obyekt ko'rsatkichlari

Mazkur bob obyekt ko'rsatkichlari bilan ishlash haqida.Keyingi sahifada siz bir sinfga mansub ko'p obyektlar bilan ishlaganda obyekt ko'rsatkichlari qanchalik muhim ekanligini ko'rasiz.

11.5.1 Obyektlarning dinamik joylashuvi

Obyektlrni xipga joylashtirish keng tarqalgan. 7.4 bo'limda ko'rib chiqilganidek, xipdan xotirani olish uchun siz yangi operatordan foydalanasiz. Masalan, chaqiruv new CashRegister ko'rsatkichni a CashRegister obyektga qaytaradi.. siz shuningdek qurilish argumentini ta'minlashiningz mumkin:

```
new BankAccount(1000)
```

siz odatda yangi operatorni qaytaruvchi ko'rsatkichni xotirada saqlashni hohlaysiz:

```
CashRegister* register_pointer = new CashRegister;
BankAccount* account_pointer = new
BankAccount(1000);
```

E'tibor bering bu tariflarning har biri 2ta modulga joylashgan. : ko'rsatkich o'zgaruvchisi va xipdagi obyekt. 8tasvirgv qarang.

Xipdagi obyekt sizga ortiq kerak bo'lmaganda , uni o'chirib tashlash uchun ishonch hosil qiling :

```
delete account_pointer;
```

11.5.2 the -> operator

Chunki registr ko'rsatkichi a CashRegister obyekti ko'rsatkichidir, the value *register_ pointer qiymati CashRegister obyektining o'zini belgilaydi To invoke a member function on that object o'sha obyektningfunksiya elementini yuklash uchun , siz chaqirishiningiz kerak.

Qavslar juda muhim, chunki C++da nuqtalar operatori operatordan yuqori turadi. qavslarsiz ifodalar kompillyasiya vaqti xatosi hisoblanadi.

*register_pointer.add_item(1.95); // xato-siz uni ko'rsatkichga qo'llay olmaysiz. nuqtalar operatori *dan yuqori bo'lganligi sababli, nuqtalar obyektga emas ko'rsatkichga qo'llaniladi.

Because calling a member function through a pointer is very commonfunksiya elementini ko'rsatkich orqali chaqirish ko'p tarqalgan bo'lganligi sababli, "ko'rsatkichni kuzat va funksiya elementini chaqir" operasiyasini

qisqartirsh uchun operator mavjud Bu operator quyidagicha yoziladi-> va odatda"strelka" deb aytiladi. bu yerda siz "strelka"operatori qanday ishlatishingiz ko’rsatilgan:

```
register_pointer->add_item(1.95);
```

bu chaqiriqning ma’nosi: Funksiya elementi qo’shimcha elementini faollashtirganda, *register_pointer ga taxminiy parametrni va 1.95. aniq parametrni o’rnating.

11.5.3 this ko’rsatkichi

Har bir funksiya elementi maxsus parametr o’zgaruvchisiga ega, bu taxminiy parametr ko’rsatkichi deb ataladi.

Masalan, CashRegister::add_item funksiyasini ko’rib chiqamiz. agar sizregister1.add_item(1.95) chaqirsangiz, unda bu ko’rsatkich CashRegister* tipiga ega bo’ladi.va obyektning the register1ini ko’rsatadi. siz bu ko’rsatkichdan funksiya elementi tarifi ichida foydalanishinigiz mumkin.

Masalan:

```
void CashRegister::add_item(double price)
{
this->item_count++;
this->total_price = this->total_price + price;
}
```

Mavzuga doir test savollari:

1. Quyidagi ifoda qanday natija chiqaradi?

```
class Base{
public:  virtual void display( int i)
  {std::cout<<"Base::"<<i;} };
class Derv: public Base{
public: void display(int j)
   { std::cout<<"Derv::"<<j; } };
int main()
{    Base *ptr = new Derv;
```

```
ptr->display(10);  return 0;}
```

a) Derv 10
b) Derv 12
c) Derv 1
d) Derv 0

2. Quyidagi ifoda qanday natija chiqaradi?

```
class sinf{
    private: int a,b;
    public: int sum(){return a+b;}
        void get(int x, int y){a=x; b=y;}
    };
int main(){
    sinf uzgaruvchi;
    uzgaruvchi.get(12,3);
    cout<<uzgaruvchi.sum()<<endl;}
```

a) 14
b) 25
c) 15
d) 11

3. Quyidagi ifoda qanday natija chiqaradi?

```
class sinf{
    private: int a,b;
    public: int sum(){return a+b;}
    };
int main(){
    sinf uz;
    uz.a=2; uz.b=3;
    cout<<uzgaruvchi.sum()<<endl;}
```

a) 1
b) 0
c) Kompilyatsiya xatoligi
d) Sintakti xatolik

4. Quyidagi ifoda qanday natija chiqaradi?

```
class sinf{
    private: int a,b;
    public: int sum(){return a+b;}
    };
int main(){
    sinf uz;
    uz.a=2; uz.b=3;
    cout<<uzgaruvchi.sum()<<endl;}
```

a) a=1 b=0 1

b) a=1 b=1 0
c) Kompilyatsiya xatoligi
d) a=0 b=0 0

5. Quyidagi ifoda qanday natija chiqaradi?

```
class TwoValues {
  int a, b;
  public:   TwoValues(int i, int j) { a = i; b = j; }
     friend class Min;  };
class Min {
  public:  int min(TwoValues x) { return x.a < x.b ? x.a : x.b; } };
int main() {
  TwoValues ob(10, 20);
  Min m;   cout << m.min(ob); return 0;
}
```

a) 10
b) 11
c) Kompilyatsiya xatoligi
d) 12

Nazorat savollari:

1. Sinf nima?
2. Sinfning qanday turlari bor?
3. Sinflarga qanday murojat qilinadi?
4. Sinf metodlari qanday murojat qilinadi?
5. Sinf obyektlariga qanday murojat qilinadi?
6. Public, private va protected nimani anglatadi?
7. Do’st sinflar qndey e’lon qilinadi?
8. Do’st sinfning vazifasi nimadan iborat?
9. Sinfning yashirin sohasidagi metodlri nima uchun ushlatiladi?
10. Qandey sinflar abstract siflar deyiladi?

12. Merosxo’rlik

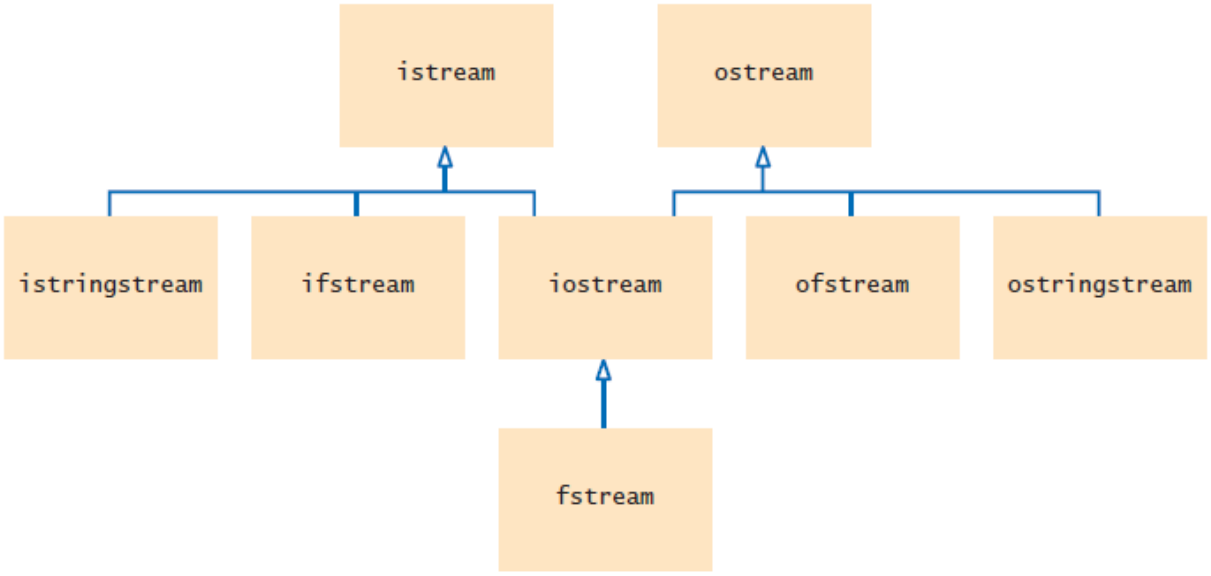
Meros va polimorfizm tushunchalarini anglash, merosga egalik qilishni va komponentlik, funksiyasini bekor qilishni o’rganish, mavjud sinfga konstruktorlarni joriy qila olish, virtual funksiyalarni ishlatish va yarata olish.

Bir biriga bog’langan sinflar odatda umumiy xarakatga egadir. Masalan, kurak, xaskash va qaychilarning barchasi bog’dagi ishlarni qilishga mo’ljallangan. Ushbu bobda, siz merosxo’rlik tushunchasining maxsus va umumiy sinflar o’rtasidagi munosabatlarni qanday ifodalashini o’rganasiz. Merosxo’rlikni ishlatish orqali, sinflar o’rtasidagi kodni bo’lishishingiz, hamda bir necha sinflar uchun xizmat ko’rsatishiniz mumkin.

12.1 Merosxo’rlik ierarxiyasi

Obyektga yo’naltirilgan sxemada, merosxo’rlik bu umumiy sinf (asosiy sinf) va maxsus sinf(quyi sinf) o’rtasidagi munosabatdir. Yordamchi sinf ma’lumot va aniqlanmagan xarakatlarni asosiy sinfdan meros qilib oladi. Masalan, 1-shaklda tasvirlangan turli xil transport vositalarining munosabalariga ahamiyat bering.Đu1040?vtomobillar barcha transport vositalari uchun umumiy bo’lgan xususiyatga, ya’ni yo’lvchilarni bir joydan ikkinchi joyga yetkazish vazifasini bajarmoqdalar. Bu biz Avtomobil sinfi Transport sinfidan meros qilib olyapti deb aytamiz. Bu o’rinda, Transport vositasi asosiy sinf va Avtomobil sinfi quyi sinf hisoblanadi.Đu1053?orasmiy ma’noda, merosxo’rlik munosabati relationship is called the is-a rela-tionship. Bu biz 9.7 bo’limda da muxokama qilgan ega bo’lish munosabatidan farqlanadi. Har bir avtomobil transport vositasidir. Har bir transport vositasi motorga ega.Đu1052?yerosxo’rlik munosabati juda kuchlidir, chunki u algoritmlar bilan obyektlarni turli xil sinflarda qayti ishlatish imkonini beradi. Balki bizda Transport ob’ktini boshqaruvchi algoritm mavdud bo’lishi mumkin. Chunki avtomobil transport vositasining maxsus turlaridan biri bo’lib, avtomobil obyektini shunday algoritm bilan ta’minlashimiz mumkinki,u to’g’ri ishlay oladi. Bu o’rinni bosish prinsipiga doir misol bo’lib,doimo asosiy sinf obyekti kerak

bo’lgan vaqtda yordamchi sinf obyektidan foydalanishingiz mumkinligidan dalolat beradi.



Merosxo’rlik munosabati 1 shaklda ko’psatilgani kabi, sinflarning ko’proq ixtisolashtirilgan ierarxiyalarni oshirira oladi. 2 shaklda ko’rsatilgan C++ sinflarining ishlash yo’li bunday ierarxiyaning yana bir misolidir. 2 shaklda merosxo’rlikdagi asosiy va quyi sinflarni asosiy sinfga yo’naltiruvchi o’q bilan birlashishida UML belgilaridan foydalanilgan. Ko’rib turganingizdek, `ifstream` (kiruchi oqim bo’lib, ma’lumotni fayldan o’qiy oladi) "istream" ning maxsus xolatidir (kiruvchi oqim bo’lib, ma’lumoni istalgan manba’dan o’qiy oladi). Agar "ifstream" ga ega bo’lsangiz, bu `"istream"`dakutilayotgan funksiya uchun argument bo’lib xizmat qiladi. void process_input(istream& in) // Can call with an ifstream object

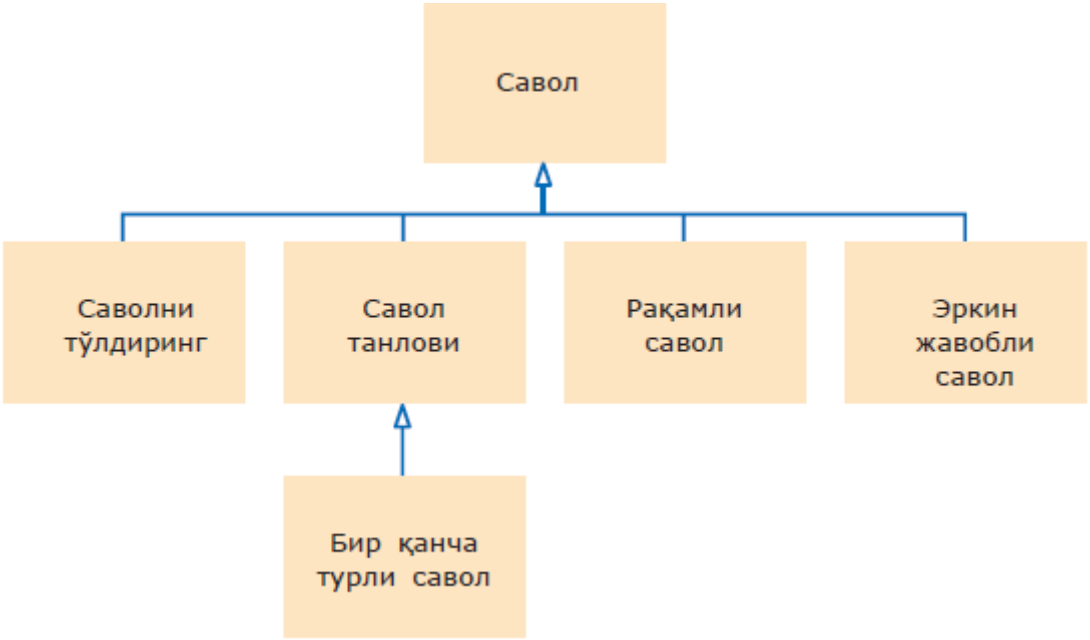
Nima uchun `ifstream` obyektlari o’rniga uning vazifasini bajaruvchi `istream` obyektlarini beramiz?

Bu funksiyaning foydasi ko’proq, chunki u har qanday kiruvchi oqim bilan ishlay oladi (masalan `istringstream` kabi, imtixon qilishga qaysi bii qulay bo’lsa). Bu yana ishdagi o’rin bosish tamoyilidir. Bu bo’limda, biz sinflarning sodda ierarxiyasini ko’rib chiqamiz. Ehtimol, kompyuterga oid testlarni ishlab borasiz. Test bir necha savollar tartibidan tashkil topgan:

- Bo’sh qatorlarni to’ldiring

• Tanlash (bir yoki bir necha)

• Sonli .



Bu ierarxiyaning o'zagida Savol turi turibdi. Savol matnni monitorda ko'sata oladi va berilgan javobning qanchalik to'g'riligini tekshiradi:

```
class Question
{
public:
Question();
void set_text(string question_text);
void set_answer(string correct_response);
bool check_answer(string response) const;
void display() const;
private:
string text;
string answer;
};
```

Matnning qanday aks etishi savol turiga bog’liq. Keyinroq bu bobda, ba’zi xilma xilliklarni ko’rib chiqasiz, lekin asosiy sinf savol matnini shundayligicha "cout" ga uzatadi. Javobning tekshirilishi ham savol turiga bog’liq. Aytib o’tilganidek, raqamli savol taxminiy javoblarni qabul qilishi mumkin. (Mashq P10.1 ga qarang). P10.3 mashqda, siz javobni tekshirilishining boshqa yo’lini ko’rasiz. Lekin asosiy sinfda, biz javobni to’g’ri javobga aynan mos tushishini talab qilamiz. Quyidagi bo’limlarda, siz asosiy sinfdan komponentlik funksiyasi va ma’lumotlar elementini meros qilib oluvchi yordamchi sinflarni qanday qilib xosil qilishni ko’rib chiqasiz. Bu yerda Savol sinfining joriy qilinishi va sodda test dasturi berilgan. Shuni nazarda tutingki, Savol sinfi konstruktori hyech qanday ishni amalga oshirishi kerak emas, chunki matn va javob maydonlari avtomatik ravishda bo’sh qatorga joylashtirilgan. "boolalpha" oqimi boshqaruvchisi asosiy vazifasi orqali Bulev (mantiqiy) qiymatini keltirib chiqarib, 1 yoki 0 ni o’rniga to’g’ri yoki noto’g’ri bo’lib ko’rsatilmoqda.

12.2 Quyi sinflarning qo’llanilishi

C++ da, siz quyi sinfni asosiy sinfdan uni qanday xususiyat o’zgartirib ko’rsatishiga qarab xosil qilasiz. Quyi sinf uchun yangi bo’lgan komponentlik funksiyalarini aniqlaysiz. Quyi sinf komponentlik funksiyasining barchasini asosiy sinfdan meros qilib oladi, lekin agar meros qilib olingan xarakat noto’g’ri bo’lsa, uning joriy qilinishini o’zgartirishingiz mumkin. Quyi sinf ma’lumotlar elementining barchasini asosiy sinfdan avtomatik ravishda meros qilib oladi. Siz faqatgina ma’lumotlar elementini aniqlaysiz. Quyida quyi sinf tafsifi uchun sintaksis berilgan:

```
class ChoiceQuestion : public Question
{
public:
private:
```

```
};
```

MUSTAQ Limuzin ishlab chiqaruvchisi kabi, ya’ni ishni odatiy mashinadan boshlab, uni ko’rinishini o’zgartirgani kabi, dasturchi ham quyi sinfni boshqa sinfni o’zgartirish hisobiga hosil qiladi.

Ushbu belgi : merosxo’rlikni anglatadi . Zaxiradagi public so’zi texnik sabablar uchun talab etiladi (449 betdagi 10.1 dagi Odatiy xatoga qarang).

SavollarVarianti obyekti Savol obyektidan uchta jihatdan farq qiladi:

• Undagi obyektlar javob uchun turli variantlarni saqlaydi.

• Bu yerda boshqa variantni qo’shish uchun komponentlik funksiyasi mavjud.

• Javob beruvchi tanlab olishi uchun, SavollarVarianti sinfi Monitor

funksiyasi ushbu variantlarni ko’rsatadi.

Qachonki, SavollarVarianti Savol sinfidan meros olganda, ushbu uch tafovutni xosil qilishi talab etiladi.

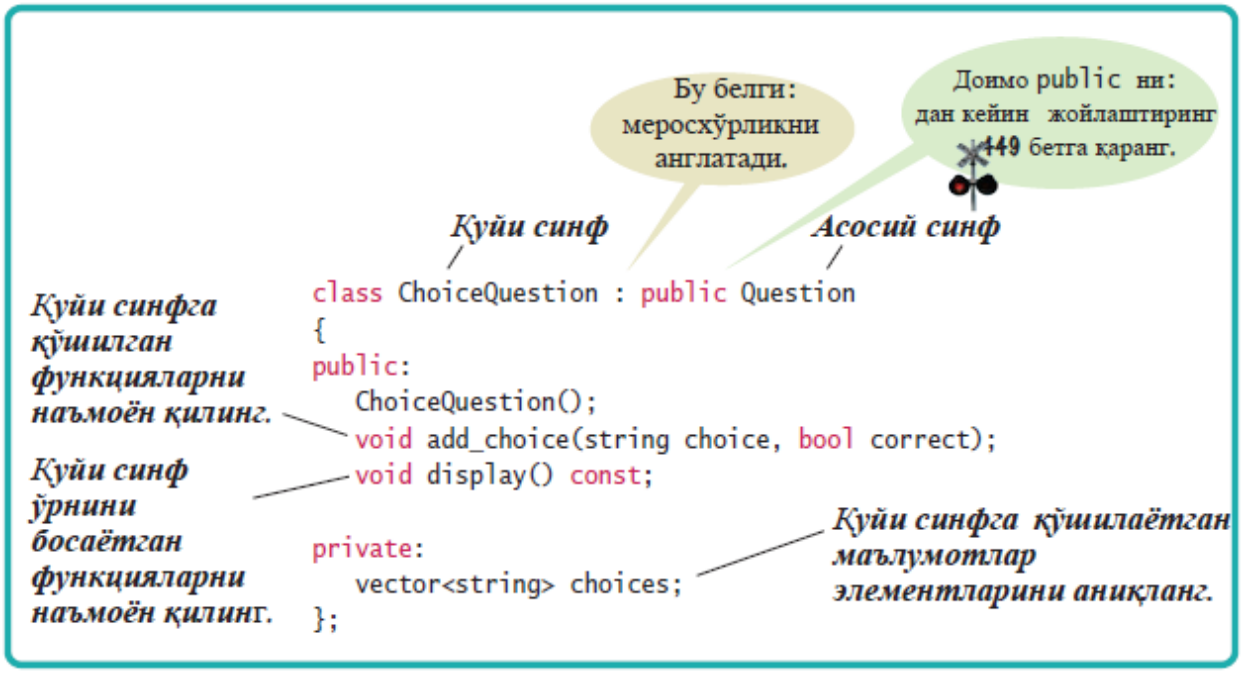
```
class ChoiceQuestion : public Question
{
public:
ChoiceQuestion( );
void add_choice(string choice, bool correct);
void display() const;
private:
vector<string> choices;
};
```

4 -shaklda SavollarVarianti obyektining joylashuvi tasvirlangan. U Savolning asosiy sinfidan ma’lumotlarning matn va javob elementlarini meros qilib olmoqda, shuningdek qo’shimcha ma’lumotlar elementini: variantlar vektorini qo’shmoqda.

add_choice funksiyasi S avo l l aV a ri anti sinfig a xo s.

Siz buni faqatgina S avo l l arV a rianti obyektlari uchun qo'llay olasiz, umumiy Savo l obyektlari uchun emas. Shunga qaramay, m onitor funksiyasi asosiy sinfda mavjud bo'lgan funksiyaning boshqa ko'rinishi bo'lib, yordamchi sinfga kerak bo'lgan maxsus zaruriyatlarni qamrab olgan. Bu xolatda biz yordamchi sinf bu funksiyani o'rnini bosyapti deb aytamiz. Buning qanday bo'lishini 10.3 -bo'limda ko'rasiz. Savollar V arianti sinfi tavsifida siz faqat yangi komponentlik funksiyalarini va ma'lumotlar elementini aniqlaysiz. Savollar sinfidagi qolgan barcha komponentlik funksiyalari va ma'lumotlar elementlari Savollar sinfi orqali avtomatik ravishda meros qilib olinadi. Masalan, har bir Savolla r Varianti obyekti hamon matn va javob ma'lumotlar elemetiga egadi r, shuningdek set_text,

set_answer, va check_answer komponentlik funksiyalari ham mavjuddir.



Siz quyi sinf obyektida meros qilib olingan komponentlik funksiyalarini quyidagicha atashingiz mumkin:

```
choice_question.set_answer("2");
```

Shunga qaramay, meros qilib olingan ma'lumot elementlardan foydalanib bo'lmaydi. Chunki bu elementlar asosiy sinfning shaxsiy ma'lumotlari bo'lib,

faqatgina asosiy sinf ulardan foydalana oladi. Yordamchi sinf boshqa sinflardan ortiqroq imkoniyatga ega emasdir.

Asosan, SavollarVarianti komponentlik funksiyasi to'g'ridan to'g'ri javob elementidan foydalana ololmaydi. Ushbu komponentlik funksiyalari Savol sinfining shaxsiy ma'lumotlari va boshqa har qanday funksiyalaridan foydalanishlari uchun ochiq interfeysdan foydalanishlari shart. Buni ko'rsatib berishimiz uchun, add_choice komponentlik funksiyasini qo'llashimizga ijozat bering. Funksiya ikki parametrga ega: variant qo'shilishi mumkin (Bunda, variantlar vektoriga qo'shimcha bo'ladi) va Bulev qiymati tanlov to'g'ri yoki noto'g'ri ekanligini bildiradi. Agar to'g'ri bo'lsa, javobni joriy variant raqamiga qo'ying. (Biz ostringstream dan raqamni chiziqqa o'tkazishda foydalanamiz

```
void ChoiceQuestion::add_choice(string choice, bool
correct)
{
choices.push_back(choice);
if (correct)
{
ostringstream stream;
stream << choices.size();
string num_str = stream.str();
...
}
}
```

Siz javob elementidan faqatgina asosiy sinfning o'zida foydalanmaysiz Yaxshiyamki, Savol sinfi set_answer komponentlik funksiyasiga ega. Siz ushbu funksiyani yordamga chaqirishingiz mumkin. Qaysi obyekt uchunt? Siz hozirgina o'zgartirayotgan savolingiz, ya'ni SavollarVariantining yashirin

parametri:add_choice funksiyasidir. 9 bobda ko’rganingizdek, agar siz kom ponentlik f unksiyasini yash irin param yetrda ish ga tush irsangiz, siz param yetrni aniqlam asdan turib, f aqatgina kom ponentlik f unksiyasi nomini yozasiz:

```
set_answer(num_str);
```

Kompilyator buni quyidagicha na’m oyon etadi:

```
implicit param eter.set_answer(num_str);
```

12.2.1 Shaxsiy merosxo’rlik

Yordamchi sinf nomidan keyin turadigan ikki nuqtadan so’ng qo’yiladigan zaxiradagi public so’zining esdan chiqishi odatiy xatodir.

```
class ChoiceQuestion : Question // Xatolik
{
...
};
```

Sinf tavsifi uzatiladi. ChoiceQuestion hamon Question dan meros oladi, lekin norasmiy tarzda meros oladi. Ya’ni, ChoiceQuestion dagi komponentlik funksiyasining o’zi Question dagi komponentlik funksiyasini chaqira oladi. Qachonki boshqa funksiya Question komponentlik funksiyasi obyektini faollashtirsa, kompiller xato bo’lganligi haqida belgi beradi:

```
int main()
{
ChoiceQuestion q;
...
cout << q.check_answer(response); // Xatolik
}
```

Bunday shaxsiy merosxo’rlikning foydasi kamdan kam tegadi. Aslida, birinchi galda bu merosxo’rlikning ishlatilish ruxini buzadi,—ya’ni, asosiy sinf

obyektlari kabi qulay obyektlarni yaratadi. Siz doimo ochiq merosxo'rlikdan foydalanib, yordamchi sinf tavsifini zaxiradagi public so'zi bilan ta'minlashni unutmasligingiz kerak.

12.2.2 Asosiy sinf komponentlaridan nusxa olish

Yordamchi sinf asosiy sinf ma'lumotlar elementidan foydalana ololmaydi.

```
ChoiceQuestion::ChoiceQuestion(string
question_text)
{
text = question_text; // Xatolik
}
```

Kompilyator xato qilganda, bu muammoni yechish uchun odatda boshlovchilar ayni nomli boshqa ma'lumot elementini yordamchi sinfga biriktiradilar:

```
class ChoiceQuestion : public Question
{
...
private:
vector<string> choices;
string text; // mumkin emas!
}
```

Albatta, konstruktor dasturni tuzadi, lekin u to'g'ri matnni o'rnatmaydi! Bu kabi ChoiceQuestion obyekti ikki xil ma'lumot elementiga ega, ikkisi ham matn deb nomlanadi. Konstruktor birini o'rnatadi va namoyish komponentlik funksiyasi ikkinchisini namoyish qiladi.

Asosiy sinf ma'lumot elementidan bexuda nusxa olgandan ko'ra, asosiy sinf komponentini yangilovchi komponentlik funksiyasiga murojat qilishingiz darkor,

misolda keltirilgan set_text funksiyasi kabi.

Qiymatlar Uchun Yagona Sinf O’zgarishidan foydalanish, Harakatdagi O’zgarish Merosxo’rligi Merosxo’rlikning maqsadi turli xarakatdagi obyektlarni modellashtirishdir.

Talabalar merosxo’rlikni ilk bor o’rganganlarida, uni xaddan tashqari ishlatishga moyildirlar. O’zgarishlar oddiy ma’lumotlar elementi yordamida na’moyon bo’lishiga qaramay, bir necha sinflarni yaratadilar. Avtoparkda yoqilg’i ishlatilish samaradorligini kuzatuvchi bir dasturni ko’rib chiqsak:bosib o’tilgan masofa va quyilgan yoqilg’i miqdorini qayd qilamiz. Parkdagi ba’zi avtomobillar gibridlardir. Siz HybridCar xosila sinfini yaratmoqchimisiz? Bu dasturiy vositada emas. Gibridlar xaydashda va yoqilg’i quyishga kelganda boshqa avtomobillardan farq qilmaydilar.

Ular faqat yaxshiroq yoqilg’i samaradorligiga egadirlar. Yagona avtomashina sinfidagi ma’lumot elementi:

```
double miles_per_gallon;
```

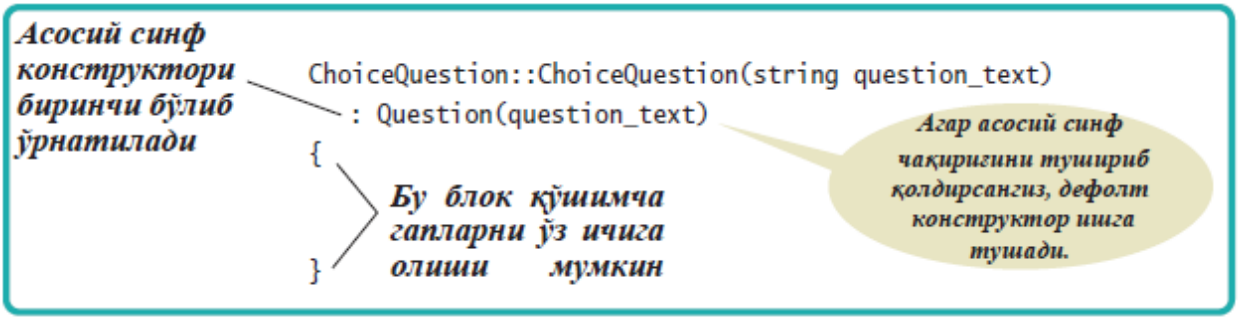
butunlay yetarlidir. Lekin, agar siz turli avtomobillarni qanday tuzatish kerakligini ko’rsatuvchi dastur yaratmoqchi bo’lsangiz, shundagina HybridCar aloxida sinfini tashkil qilish ma’no kasb etadi. Chunki tuzatishga kelganda, gibrid mashinalar boshqa avtomobillardan farqli harakatga egadirlar.

Asosiy sinf konstruktoriga murojaat qilish Yordamchi sinf obyektini yaratish jarayonini tasavur qiling. Yordamchi sinf kostruktori faqatgina ushbu sinf ma’lumotlar elementini avvalgi xolatiga qaytarishi mumkin. Lekin asosiy sinf ma’lumotlar elemeti ham avalgi xolatga qaytarilishni talab etadi. Biroq siz aksini qilsangiz, asosiy sinf a’lumotlar elementi asosiy sinfning mavjud konstruktori orqali avvalgi xolatga qaytariladilar. Boshqa konstruktorni aniqlash uchun, 9.1-maxsus mavzuda tasvirlangani kabi, inisializator ro’yxatini ishlating. Asosiy sinf nomini va inisializator ro’yxatidan konstruksiya parametrlarini aniqlang. Masalan, Question asosiy sinfi savol matnini o’rnatuvchi konstruktorga ega bo’lgan deb hisoblang. Bu yerda yordamchi sinf konstruktori asosiy sinf konstruktoriga qanday

murojaat qilishi ko’rsatilgan.

```
ChoiceQuestion::ChoiceQuestion(string
question_text)
: Question(question_text)
{
}
```

Yordamchi sinf konstrutori { } ichidagi kodni bajarishdan oldin murojaat qiladi. Na’muna qilib berilgan dasturimizda, biz asosiy sinfning defolt konstruktordan foydalandik. Shunga qaramay, agar asosiy sinfda defolt konstruktor bo’lmasa, inisiyalizator ro’yxati sintaksisidan foydalanishiingiz zarur.



12.3 Komponentlik funksiyalarini bekor qilish

Yordamchi sinf komponentlik funksiyalarini asosiy sinfdan meros qilib oladi. Agar siz meros qilib olingan komponentlik funksiyasi harakatidan qoniqmasangiz, yordamchi sinfga yangilik kiritish orqali, uni bekor qilishingiz mumkin.

ChoiceQuestion sinfidagi display funksiyasini ko’rib chiqing. Asosiy sinfning display funksiyasi, javob variantlarini taqdim etish maqsadida bekor qilinishi lozim. Asosan, asosiy sinf funksiyasi:

• Savol matnini ko’rsatishi.

• Javob variantlarini ko’rsatishi kerak bo’ladi.

Ikkinchi qismi oson xisoblanadi, chunki javob variantlari yordamchi sinfning ma’lumot elementidir.

```
void ChoiceQuestion::display() const
{
for (int i = 0; i < choices.size(); i++)
{
cout << i + 1 << ": " << choices[i] << endl;
}
}
```

Lekin siz qanday qilib savol matnini olasiz? Siz asosiy sinfning matn elementidan to’g’ridan to’g’ri foydalana olmaysiz, chunki u yopiqdir. Buning o’rniga, asosiy sinfning display funksiyasini chaqirasiz.

```
void ChoiceQuestion::display() const
{
display();
...
}
```

Shunga qaramay, bu yaxshi ish bermaydi. Chunki ChoiceQuestion::display dagi yashirin parametr ChoiceQuestion turiga mansub bo’lib, ChoiceQuestion sinfida display deb nomlanuvchi funksiyaga ega. Bu funksiya chaqiriladi—lekin bu siz ayni vaqtda yozgan funksiyangizning o’zidir! Bu funksiya o’zini o’zi qayta qayta chaqiraveradi.

Savol matnini ko’rsatish uchun, chaqirmoqchi bo’lgan display deb nomlanuvchi funksiyaning qaysi birini tanlashni aniqlashingiz shart. Sizga `Question::display:` kerak:

```
void ChoiceQuestion::display() const
{
Question::display(); // OK
```

```
...
}
```

Siz funksiyani bekor qilganingizda, odatda asosiy sinf versiyasining funksionalligini oshirishga urinasiz. Shuning uchun, asosiy sinf versiyasini kengaytirishdan oldin, uni faollashtrishingiz darkor. Uni faollashtirish uchun, BaseClass::function notasiyasidan foydalanishingiz lozim. Ammo, asosiy sinf funksiyasini chaqirishingiz majburiy emas. Ba'zida, yordamchi sinf asosiy sinf funksiyasini bekor qilib,butunlay boshqa funksionallikni aniqlaydi. Quyida Question obyekti va ChoiceQuestion obyekti aqiq ko'rsatilgan to'liq dastur berilgan. (Siz yuqorida ko'rgan Question sinfining tavsifi question.h ga joylashtirilgan va question.cpp. ga joriy qilingan) Bu misol asosiy sinfdan maxsus sinfni shakllantirishda merosxo'rlikning ishlatilishi ko'rsatilgan.

```
#include <iostream>
 #include <sstream>
#include <vector>
#include "question.h"
class ChoiceQuestion : public Question
{
public:
void add_choice(string choice, bool correct);
void display() const;
private:
vector<string> choices;
};
ChoiceQuestion::ChoiceQuestion(){
}
```

```
void ChoiceQuestion::add_choice(string choice, bool correct)
     {
     choices.push_back(choice);
     if (correct) {
     ostringstream stream;
     stream << choices.size();
     string num_str = stream.str();
     set_answer(num_str); }
     }
     void ChoiceQuestion::display() const{
      Question::display();
      for (int i = 0; i < choices.size(); i++)
     {
     cout << i + 1 << ": " << choices[i] << endl;
     } }
     int main()
     string response;
     cout << boolalpha;
    Question q1;
     q1.set_text("Who was the inventor of C++?");
     q1.set_answer("Bjarne Stroustrup");
     q1.display();
     cout << "Your answer: ";
     getline(cin, response);
```

```
      cout << q1.check_answer(response) << endl;
      ChoiceQuestion q2;
q2.set_text("In which country was the inventor of C++ born?");
      q2.add_choice("Australia", false);
      q2.add_choice("Denmark", true);
      q2.add_choice("Korea", false);
      q2.add_choice("United States", false);
      q2.display();
      cout << "Your answer: ";
      getline(cin, response);
      cout << q2.check_answer(response) << endl;  return 0;
      }
```

Natija:

```
Who was the inventor of C++?
Your answer: Bjarne Stroustrup
true
In which country was the inventor of C++ born?
1: Australia
2: Denmark
3: Korea
4: United States
Your answer: 2
true
```

Asosiy sinf nomini unutish asosiy sinf funksiyasi funksionalligini oshirishdagi odatiy xato bu asosiy sinf nomini unutishdir. Masalan, menedjerning

maoshini hisoblaganda, maoshni Employee obyektining asosidan oling va bonus qo’shing.

```
double Manager::get_salary() const
{
double base_salary = get_salary();
return base_salary + bonus;
}
```

Bu yerda get_salary() get_salary funksiyasiga tegishli bo’lib, komponentlik funksiyasining yashirin parametri uchun qo’llanilgan.Yashirin parametr Manager turiga xos bo’lib, bu yerda Manager::get_salary funksiyasi mavjud vabu funksiya chaqiriladi. Albatta, biz yozgan funksiyaning rekursiv chaqirig’idir. Buning o’rniga, siz qaysi get_salary funksiyasini chaqirmoqchiligingizni aniqlashtiring. Bunday g’olatda, siz ochiq ravishda Employee::get_salary funksiyasini chaqirishingiz kerak.

Harqachon, yordamchi sinf funksiyasidan asosiy sinf funksiyasini bir xil nom bilan chaqirganingizda, funksiyaga asosiy sinf nomini hisobga olgan holda to’liq nom berganingizga amin bo’ling.

12.4 Virtual funksiyalar va polimorfizm

Oldingi bo’limlarda merosxo’rlikdan foydalanishning muxim jixatlarini ko’rdingiz: asosiy sinfdan maxsus sinfni xosil qilish. Quyidagi bo’limlarda merosxo’rlikning yanada kuchliroq dasturiy vositalarini ko’rib chiqasiz: dastur ishlayotgan vaqtda harakati va turi o’zgaruvchi obyektlar bilan ishlaysiz. Bunday harakat xilma xilligiga **virtual funksiyalar** tufayli erishiladi. Obyektda virtual funksiyani faollashtirganingizda, C++ dastur tizimi aynan qaysi komponentlik funksiyasini chaqirishni aniqlaydi, bu obyektning qanday sinfga tegishli ekanligiga bog’liqdir. Quyidagi bo’limlarda, nima uchun sinfi dastur ishlayotgan vaqtda o’zgaruvchi obyektlardan foydalanishda ko’rsatkichlarni ishlatish kerakligini

bilasiz, va qanday qilib virtual funksiyalar keltirilgan obyektga mos keluvchi komponentlik funksiyasini tanlashini ko’rib chiqasiz.

12.4.1 Muammoni kesib tashlash

Bu bo’limda, sinf ierarxiyasida turli xil sinflarga tegishli obyektlar yig’indisi bilan ishlaganda yuzaga keladigan odatiy muammoni muxokama qilamiz. Agar siz quiz2/test.cpp ning asosiy funksiyasiga nazar tashlasangiz, u yerda savolni ko’rsatuvchi va javoblarni tekshiruvchi bir necha marotaba qaytarilayotgan kodlarni uchratasiz. Agar barcha savollar bir massivda joylashtrilgan bo’lib, ularni foydalanuvchiga taqdim etishda takrorlashdan foydalanilsa maqsadga muvofiq bo’lar edi:

```
const int QUIZZES = 2;
Question quiz[QUIZZES];
quiz[0].set_text("Who was the inventor of C++?");
quiz[0].set_answer("Bjarne Stroustrup");
ChoiceQuestion cq;
cq.set_text("In which country was the inventor of
C++ born?");
cq.add_choice("Australia", false);
...
quiz[1] = cq;
for (int i = 0; i < QUIZZES; i++)
{
quiz[i].display();
cout << "Your answer: ";
getline(cin, response);
cout << quiz[i].check_answer(response) << endl;
```

```
}
```

Massivdagi quiz Question turidagi obyektga ega. Kompilyator ChoiceQuestion Questionningmaxsus xolati ekanligini anlaydi. Shuning uchun, u variantlar savolini savoldan o’zlashtirib olishga ruzsat beradi:

```
quiz[1] = cq;
```

Biroq, ChoiceQuestion obyekti usta ma’lumotlar elementiga ega, Question obyekti esa faqatgina ikkita ega. Yordamchi sinf ma’lumotini saqlash uchun xona mavjud emas. Bu ma’lumot, asosiy sinf o’zgaruvchisidan yordamchi sinf obyektini o’zlashtirganingizda shunchaki kesib tashlanadie (5 -shaklga qarang). Agar natijani dasturda qo’llasangiz, variantlar ko’rsatib berilmaydi :

```
C++ Asoschisi kim?
Sizning javobingiz: Bjarne Stroustrup
true
C++ asoschisi qayerda tug’ilgan?
Sizning javobingiz:
```

Bu xolat, merosxo’rlik ierarxiyasida sinflar aralashmasidagi obyektni boshqarmoqchi bo’lgan kodga o’ta xos muammodir. Quyi sinf obyektlari odatda asosiy sinf obyektlaridan kattaroq bo’lib, turli yordamchi sinf obyektlari turli o’lchamga egadir. Obyektlar massivi bu o’lchamlar xilma xilligi bilan ish ko’ra olmaydi.

Buning o’rniga, boshqa joylardagi real obyektlarni saqlashingiz va ularning joylashuvlarini, ko’rsatkichlarni saqlash orqali massivga to’plashingiz kerak. Biz ko’rsatkichlarning ishlatilishini keyingi bo’limda mug’okama qilamiz.

12.4.2 Asosiy va yordamchi sinf ko’rsatkichlari

Sinf ierarxiyasida, turli sinflardagi obyektlardan foydalanish uchun, ko’rsatkichlarni ishlating. Turli obyektlar ko’rsatkichlarining barchasi yagona o’lchamga ega —ya’ni, xotira adresi o’lchami— shunga qaramay, obyektlarning

o’zi turli o’lchamlarga ega bo’lishlari mumkin.

Bu yerda ko’rsatkichlar massisini o’rnatish uchun kod berilgan (6-shaklga qarang):

```
Question* quiz[2];
quiz[0] = new Question;
quiz[0]>
set_text("Who was the inventor of C++?");
quiz[0]>
set_answer("Bjarne Stroustrup");
ChoiceQuestion* cq_pointer = new ChoiceQuestion;
cq_pointer>
set_text("In which country was the inventor of C++
born?");
cq_pointer>
add_choice("Australia", false);
...
quiz[1] = cq_pointer;
```

Ajratib qo’yilgan kod ko’rsatib turganidek, siz massivni ko’rsatkichlarni saqlash uchungina aniqlaysiz, barcha obyektlarni yangisini new chaqirish orqali o’rnating, va nuqta operatori o’rniga > operatorini ishlating. E’tiborlisi shundaki, oxirgi o’zlashtirma ChoiceQuestion* tipidagi yordamchi sinf ko’rsatkichini, asosiy sinf Question* tipidagi ko’rsatkichga biriktiradi. Bunin uddasidan chiqish mutlaqo oson. Ko’rsatkich obyektning boshlang’ich adresidir. Chunki har bir ChoiceQuestion "Question" ning maxsus xolati bo’lib, ChoiceQuestion obyektinig boshlang’ich adresi, asosan, Question obyektining boshlang’ich adresidir. Qarama qarshisi—ya’ni asosiy sinf ko’rsatkichidan yordamchi sinf ko’rsatkichini o’zlashtirish —xatodir. Barcha savollarni taqdim etuvchi kod bu:

```
for (int i = 0; i < QUIZZES; i++)
{
quiz[i]>
display();
cout << "Your answer: ";
getline(cin, response);
cout << quiz[i]>
check_answer(response) << endl;
}
```

12.4.3 Virtual funksiyalar

Sinf ierarxiyasidgi turli siflarning obyektlarini to’plaganingizda, va komponentlik funksiyasini faollashtirganingizda, to’g’ri bo’lgan komponentlik funksiyasini qo’llanillishini istaysiz. Masalan, display komponentlik funksiyasini Question* ko’rsatkichi uchun chaqirganingizda, bu ChoiceQuestionga ishora qilinganda sodir bo’lib, variantlar namoyish etilishini xoxlaysiz.

Samaradorlik nuqtai nazaridan, bu defolt C++ emas. Defolt :

```
quiz[i]>
display();
```

chaqirig’i doimo Question::displayni chaqiradi,chunki quiz[i] turi Question*ga oid.

Shunga qaramay, bu xolatda siz haqiqatdan ham quiz[i]ga ishora qiluvchi

obyektning muhim turini aniqlashni istaysiz, u yo Question yoki ChoiceQuestion obyekti bo’ladi, keyin esa to’g’ri keladigan funksiyani chaqirayerasiz. C++da, siz kompilyatorni, funksiyaning chaqirilishi mos keluvchi funksiya tanlovidan oldin bo’lishi kerakligi haqtda ogohlantirishingiz shart, bu esa takrorlashdagi har bir iterasiya uchun turlichadir. Bu maqsadda, virtual yashirin so’zni ishlating:

```
class Question
{
public:
Question();
void set_text(string question_text);
void set_answer(string correct_response);
virtual bool check_answer(string response) const;
virtual void display() const;
private:
...
};
```

virtual yashirin so’z asosiy sinfda ishlatilishi kerak. Yordamchi sinfda bir nom va turli parametr turlarigi ega bo’lgan funksiyalarning barchasi keyinchalik avtomatik ravishda virtul xolatga o’tadi. Biroq, virtual yashirin so’zni yordamchi sinf funksiyalarida ham qo’llash yaxshi hisoblanadi.

```
class ChoiceQuestion : public Question
{
public:
ChoiceQuestion();
void add_choice(string choice, bool correct);
virtual void display() const;
private:
...
};
```

virtual yashirin so’zni funksiya tavifida ishlatmang:

```
    void Question::display() const // No virtual
reserved word
    {
    cout << text << endl;
    }
```

Virtual funksiya chaqirilgan har qanday vaqtda, kompilyator dastur bajarilayotgan vaqtda o'ziga xos chaqiriqning yashirin parametr turini aniqlaydit. So'ngra obyektga mos keluvchi funksiya chaqiriladi. Masalan, display funksiyasi virtual deb e'lon qilinganda, chaqiriq

```
quiz[i]>
display();
```

doimo obyektning aktual turiga tegishli bo'lgan funksiyani chaqiradi, bunda display[i]—yoki Question::displayga yo bo'lmasa, ChoiceQuestion::displayga ishora qiladi.

12.4.4 Ko'p shakllilik(polimorfizm)

Quiz massivi savollarning ikkisi aralashmasini to'playdi. Bunday to'plam polymorphic deb nomlanadi (ya'ni, "bir qancha shakllar" degani). Polimorfik to'plamdagi obyektlar bir oz o'xshashlikka ega bo'lsalarda, bir turga mansub bo'lishlari shart emas. Merosxo'rlik ushbu o'xshashlikni ifodalash uchun ishlatiladi,va virtual funksiyalar harakatdagi variasiyalarga imkon beradi. Virtual funksiyalar dasturlarga o'ta qayishqoqlik xususiyatini beradilar. Savol taqdimoti takrorlashi faqatgina umumiy mexanizmni tasvirlaydi: "Savolni namoyish eting, javob oling, va uni tekshiring". Har bir obyekt, maxsus vazifalarni qanday bajarishni o'zi biladi: "Savolni namoyish eting" va "Javobni tekshiring". Virtual funksiyalardan foydalanish dasturlarni kengaytirishga onsonlik yaratadi. Tasavvur qiling, bizga hisob kitob uchun taxminiy javob olishimiz mumkin bo'lgan yangi savol turi kerak. Qilishimiz kerak bo'lgan narsa bu- o'zinig check_answer funksiyasiga ega bo'lgan NumericQuestion yangi sinfini o'rnatishdir. So'ngra quiz

massivini aniq savollar, variantli savollarva raqamli savollar aralashmasi bilan to’ldirishimiz mumkin. Savollarni namoyish qiluvchi kod umuman o’zgartirilmasligi kerak! Virtual funksiyalardagi chaqiriqlar avtomatik ravishda yangi o’rnatilgan sinflardan to’g’ri komponentlik funksiyalarini tanlab oladi. Bu yerda test dasturining oxirgi versiyasi berilgan bo’lib, unda ko’rsatkichlar va virtual funksiyalar ishlatilgan. Dasturni ishlatganingizda, virtual funksiyalarga mos keluvchi versiyalar chaqirilganligini ko’rasiz. (question.cpp va choicequestion.cpp fayllari kitobingizning ikkinchi qismidagi kodlarga kiritilgan.)

question.h fayli

```
#ifndef QUESTION_H
 #define QUESTION_H
 #include <string>
 using namespace std;
 class Question
{
 public:
Question();
void set_text(string question_text);
virtual void display() const;
 private:
 string text;
 string answer;
 };
 #endif
```

choicequestion.h fayli

```
#ifndef CHOICEQUESTION_H
#define CHOICEQUESTION_H
#include <vector>
#include "question.h"

class ChoiceQuestion : public Question
{
public:
ChoiceQuestion();
void add_choice(string choice, bool correct);
virtual void display() const;
private:
vector<string> choices;
};
#endif
```

test.cpp fayli

```
#include <iostream>
#include "question.h"
#include "choicequestion.h"
int main()
{
string response;
cout << boolalpha;
const int QUIZZES = 2;
```

```
Question* quiz[QUIZZES];
quiz[0] = new Question;
quiz[0]->set_text("Who was the inventor of C++?");
quiz[0]->set_answer("Bjarne Stroustrup");
ChoiceQuestion* cq_pointer = new ChoiceQuestion;
cq_pointer->set_text("In which country was the inventor of C++ born?"); 20
cq_pointer->add_choice("Australia", false);
cq_pointer->add_choice("Denmark", true);
cq_pointer->add_choice("Korea", false);
cq_pointer->add_choice("United States", false);
quiz[1] = cq_pointer;
 for (int i = 0; i < QUIZZES; i++){
 quiz[i]->display();
 cout << "Your answer: ";
 getline(cin, response);
 cout << quiz[i]->check_answer(response) << endl; 33 }
return 0;
}
```

Tasavvur qiling quyidagi funksiyani Question sinfiga qo’shamiz:

```
void Question::ask() const
{
display();
cout << "Your answer: ";
getline(cin, response);
```

```
    cout << check_answer(response) << endl;
    }
```

Endi esa ushbu chaqiriqni ko’ring:

```
    ChoiceQuestion cq;
    cq.set_text("In which country was the inventor of
C++ born?");
    ...
    cq.ask();
```

Qaysi displey va check_answer funksiyasi funksiya chaqairig’ini so’raydi? Agar siz Question::ask functionsi kodining ichiga qarasangiz, bu funksiyalarning yashirin parametr bo’yicha amalga oshiralayotganini ko’rasiz:

```
    void Question::ask() const
    {
    implicit parameter.display();
    cout << "Your answer: ";
    getline(cin, response);
    cout << implicit parameter.check_answer(response)
<< endl;
    }
```

Bizning chaqirig’imizdagi yashirin parametr cq dir, ChoiceQuestion turi obyekti. Display va check_answer funksiyalari virtual bo’lganlari uchun, ChoiceQuestion versiyasidagi funksiyalar avtomatik ravishda chaqiriladi. Bu Question sinfiga savol funksiyasi o’rnatilganda ham sodir bo’laveradi, chunki u ChoiceQuestion sinfi haqida bilimga ega emas.

Ko’rib turganingizdek, virtual funksiyalar juda kuchli mexanizmdir. Savol sinfi savol so’rash funksiyasini savol so’rashga xos bo’lgan tabiat bilan

ta’minlaydi, ya’ni uni namoyish etish va javobni tekshirish. Namoyish qilish va tekshirishni qanday amalga oshirish esa yordamchi sinf vazifasidir.

Mavzuga doir test savollari:

1. Quyidagi ifoda qanday natija chiqaradi?

```
class myclass{
   int a, b;
   public:  friend int sum(myclass x);
            void set_ab(int i, int j) { a = i; b = j; } };

int sum(myclass x) {
     return x.a + x.b;    }
int main() {
     myclass n;   n.set_ab(3, 4);
     cout << sum(n);     return 0;
}
```

a) 7
b) 5
c) 9
d) 11

2. Quyidagi ifoda qanday natija chiqaradi?

```
class myclass{    int a, b;
   public: friend int max(myclass x);
            void set_ab(int i, int j) { a = i; b = j; }
     void set(){cout<<"a= "<<a<<" ";  cout<<"b= "<<b<<endl; }};
int max(myclass x) {
     if(x.a>x.b)return x.a;  else return x.b;}
int main() {
     myclass n;   n.set();
     n.set_ab(3, 4);      cout<<max(n)<<endl;
}
```

a) a=0 b=1 4
b) a=1 b=1 4
c) a=0 b=0 4
d) xatolik

3. Quyidagi ifoda qanday natija chiqaradi?

```
void f(int i){cout<<"int";}
void f(char c){cout<<"char";}
int main(){
  f('S');
}
```

a) char

b) int
c) char 1
d) xatolik

4. Quyidagi ifoda qanday natija chiqaradi?

```
inline int max(int a, int b) {
     return a>b ? a : b;}
int main() {
     cout << (20>10 ? 30:20);
     cout << " " << (99>88 ? 99:88);
}
```

a) 30 99
b) 11 22
c) 11 24
d) 99 30

5. Quyidagi ifoda qanday natija chiqaradi?

```
class Time{
  private:     int hours;
    int minutes
  public:     Time(){
      hours = 0;        minutes = 0;     }
    Time(int h, int m){
      hours = h;     minutes = m;      }
   void displayTime()     {
      cout << "H: " << hours << " M:" << minutes <<endl;     }
    Time operator++ ()   {
      ++minutes;     if(minutes >= 60)  {
        ++hours;     minutes -= 60;       }
      return Time(hours, minutes);     }
 Time operator++( int )    {
       Time T(hours, minutes);      ++minutes;
      if(minutes >= 60)   {
        ++hours;    minutes -= 60;   }
      return T;    }   };
int main()
{
  Time T1(11, 59), T2(10,40);

  ++T1   T1.displayTime();
  ++T1;  T1.displayTime();
  T2++;  T2.displayTime();
  T2++;  T2.displayTime();         return 0;
}
```

e) H: 12 M:0 H: 12 M:1
f) H: 13 M:1 H: 11 M:0
g) H: 14 M:2 H: 12 M:1
h) H: 11 M:0 H: 13 M:0

Nazorat savollari:

1. Virtual funksiyani vazifasi nima?
2. Polimorfizm nima?
3. Overload va Overide qandey jarayon v anima farqlari bor?
4. Polimorfizm qandey masalalarda ishlatiladi?
5. Abstrakt class nima va uning vazifasi nimadan iborat?
6. Do’st funksining vazifasi nima?
7. Do’st funksiya bilan do’st clasning farqi nimada?
8. Amallarni qayta yuklash nima uchun kerak?
9. Funksuyalarni qayta yuklashning vazifasi nima?
10. Classni to’liq abstrak bo’lishning qandey shartlari bor?

FOYDALANILGAN ADABIYOTLAR RO'YXATI

Asosiy adabiyotlar:

1. Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. "Voris-nashriyot" MCHJ, Toshkent 2013, 488 b.

2. Horstmann, Cay S. C++ for everyone / Cay S. Horstmann. Printed in the United States of America - 2nd ed. 2010. – P. 562.

3. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.

4. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.

5. Xaydarova M.Yu., Mallayev O.U. Visual C++ da kichik loyihalar yaratish. O'quv qo'llanma. – T.: Aloqachi, 2019. – 224 b.

Qo'shimcha adabiyotlar:

6. Стивен С. Скиена, Мигель А. Ревилла. Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям / Пер. с англ. - М: КУДИЦ-ОБРАЗ, 2005. - 416 с.

7. Меньшиков Ф. В. Олимпиадные задачи по программированию. - СПб.: Питер, 2006. - 315 с.

8. Кнут Д. Искусство программирования. Том 1-4., СПб. Вильямс 2007.

9. Холзнер С. Visual C++ 6. Учебный курс. — СПб.: Питер, 2007. - 570 с.

10. Смайли Джон. Учимся программировать на C++ вместе с Джоном Смайли. –СПб: ООО «ДиаСофтЮП», 2003.-560с.

11. Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ, 2-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2005. — 1296 с.

12. Культин Н. Б. C/C++ в задачах и примерах. — СПб.: БХВ-Петербург, 2005. -288 с.

13. Подбельский В. В., Фомин С. С. Программирование на языке Си: Учеб. пособие. - 2-е доп. изд. - М.: Финансы и статистика, 2004. - 600 с

14. Долинский М. С. Решение сложных и олимпиадных задач по программированию: Учебное пособие. — СПб.: Питер, 2006. — 366 с.

15. Павловская Т.С. Щупак Ю.С. С/С++. Структурное программирование. Практикум.-СПб.: Питер,2007-240с

16. Павловская Т.С. Щупак Ю.С. С++. Объектно- ориентированное программирование. Практикум.-СПб.: Питер, 2005-265с.

17. Романов Б.А. Практикум по программированию на С++: Учебное пособие. СПб.: BXB-Петербург, Новосибирск: Из-во НГТУ, 2006.- 432с.

Internet saytlari:

1. www.ziyonet.uz - O’zbekistan Respublikasi axborot-ta’lim portali.

2. Martijn Koster "Robots in the Web: threat or treat?". http://info.webcrawler.com/mak/projects/robots/threat-or-treat.html

3. neerc.ifmo.ru – Dasturlash bo’yicha jahon chempionatining Shimoliy Sharqiy Yevropa Mintaqasini rasmiy sayti.

4. icpc.baylor.edu - Dasturlash bo’yicha jahon chempionatining rasmiy sayti.

MUNDARIJA