O‘quv qo‘llanma

C++

T.R.Shafiyev
Sh.E.Nazarov
Z.K.Niyozova

Dasturlash asoslari fanidan

MASALALAR VA YECHIMLAR

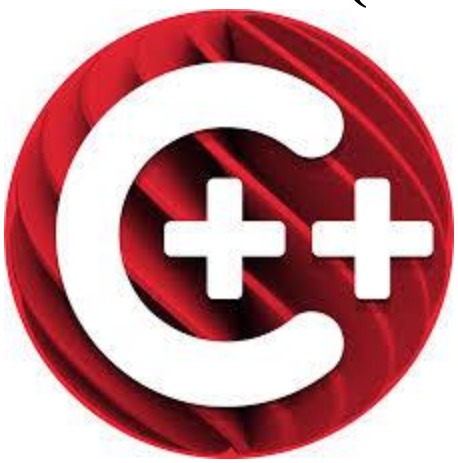
ISBN 978-991073462-5

DASTURLASH ASOSLARI FANIDAN MASALALAR VA YECHIMLAR

O‘ZBEKISTON RESPUBLIKASI
OLIY TA’LIM, FAN VA INNOVATSIYALAR VAZIRLIGI
BUXORO DAVLAT UNIVERSITETI

T.R.Shafiyev, Sh.E.Nazarov, Z.K.Niyozova

DASTURLASH ASOSLARI FANIDAN MASALALAR VA YECHIMLAR (C++)



O‘QUV QO‘LLANMA

“KAMOLOT” nashriyoti
Buxoro-2024

UO‘K: 004.424.378.1
KBK: 32.973
Sh 53

Shafiyev Tursun Rustamovich, Nazarov Shahzod Erkinovich, Niyozova Zaynabbegim Komilovna. Dasturlash asoslari fanidan masalalar va yechimlar (C++) [Matn]:o‘quv qo‘llanma / T.R. Shafiyev, Sh.E.Nazarov, Z.K. Niyozova – Buxoro: “BUXORO DETERMINANTI” Kamolot nashriyoti , 2024 - 148 b.

Ushbu o‘quv qo‘llanma “Dasturlash” bo‘yicha asosiy nazariy bilim va ko‘nikmalarni mustahkamlashga, shuning bilan birga dastur tuzish jarayonida dasturlashga oid kutubxonalar, operatorlar, matematik va mantiqiy amallar boshqa vositalarning maqsadi, vazifalari, turlari haqida ma’lumot berishga qaratilgan. Qo‘llanmada dasturlashtirishga kirish jarayonida kerak bo‘ladigan asosiy mavzular yoritilgan hamda ushbu mavzular bo‘yicha masalalar yechib ko‘rsatilgan. Mavzularni mustahkamlash maqsadida mustaqil ishlash uchun alohida topshiriqlar berilgan.

Mazkur o‘quv qo‘llanma 60540100-Matematika, 60531000-Mexanika va matematik modellashtirish ta’lim yo‘nalishlarida o‘qitiladigan “Dasturlash asoslari” fanidan talabalar foydalanishlari uchun tuzilgan.

Taqrizchilar:

F.N. Nurulloyev - Buxoro davlat universiteti “Axborot tizimlari va raqamli texnologiyalar” kafedrasi dotsenti, p.f.f.d.

U.M. Ibragimov - Buxoro muhandislik-texnologiyalari instituti “Texnologik jarayonlarni boshqarishning axborot-kommunikatsiya tizimlari” kafedrasi dotsenti, p.f.f.d.

ISBN 978-9910-734-62-0

Ushbu o‘quv qo‘llanma O‘zbekiston Respublikasi Oliy ta’lim, fan va innovatsiyalar vazirligi Buxoro davlat universitetining 2024-yil “3” fevraldagi “60”-sonli buyrug‘iga asosan ruxsat berildi.

MUNDARIJA

KIRISH

Zamonaviy dunyoda, hayotni axborot texnologiyalarisiz tasavvur qilishning iloji yo‘q. Ular bizning hayotimizga qat’iy kirishdi, axborot texnologiyalari insoniyat hayotining barcha sohalarida qo‘llaniladi, ayniqsa ikki tomonlama muhim rol o‘ynaydi. Axborot texnologiyalari insoniyatning barcha to‘plangan tajribasini amaliy foydalanish uchun mos formatlangan shaklda aks ettiradi. Shuningdek, u ijtimoiy jarayonlarni amalga oshirish uchun ilmiy bilim va materialistik tajribani jamlaydi, shu bilan birga mehnat, vaqt, energiya va moddiy xarajatlarni tejaydi.

"Bugun O‘zbekiston barcha sohalarda o‘zini namoyon qilmoqda. Axborot texnologiyalariga o‘z vaqtida e’tibor qaratishimiz yaxshi natijalar bermoqda. Bitta dasturiy mahsulot davlatga korrupsiya, byurokratiyani yo‘q qilish va odamlar uchun qulayliklar yaratishda katta yordamdir. Yodingizda bo‘lsin, sizning ishingiz juda muhim", — Shavkat Mirziyoyev.

O‘zbekiston Respublikasi Prezidentining 2021 yil 17 fevraldagi PQ-4996-sonli “Sun’iy intellekt texnologiyalarini jadal joriy etish uchun shart-sharoitlar yaratish chora-tadbirlari to‘g‘risida”gi qarori, 2021 yil 26 avgustdagi PQ-5234-sonli “Sun’iy intellekt texnologiyalarini qo‘llash bo‘yicha maxsus rejimni joriy qilish chora-tadbirlari to‘g‘risida”gi qarori, 2020 yil 5 oktabrdagi PF-6079-sonli “«Raqamli O‘zbekiston — 2030” strategiyasini tasdiqlash va uni samarali amalga oshirish chora-tadbirlari to‘g‘risida” farmoni yurtimizda Axborotlashtirish sohasiga bo‘lgan katta e’tiborni ko‘rsatadi.

Ushbu o‘quv qo‘llanma 60540100-Matematika, 60531000-Mexanika va matematik modellashtirish ta’lim yo‘nalishlarida o‘qitiladigan “Dasturlash asoslari”, fanidan talabalarning olgan bilimlarini mustahkamlash, dasturlash bo‘yicha ko‘nikmalarini oshirishga xizmat qiladi.

1-mavzu. Algoritm tushunchasi va tasvirlash usullari.

Algoritm bu oldimizga qo‘yilgan masalani yechish zarur bo‘lgan amallar ketma-ketligidir.

Algoritm so‘zi va tushunchasi IX asrda yashab ijod etgan buyur alloma Muhammad al-Xorazmiy nomi bilan uzviy bog‘liq. Algoritm so‘zi Al-Xorazmiy nomini Yevropa olimlari tomonidan buzib talaffuz qilinishidan yuzaga kelgan. Al-Xorazmiy birinchi bo‘lib o‘nlik sanoq sistemasining tamoyillarini va undagi to‘rtta amallarni bajarish qoidalarini asoslab bergan.

Algoritmning asosiy xossalari. Algoritmning 5-ta asosiy xossasi bor:

Diskretlilik (Cheklilik). Bu xossaning mazmuni algoritmlarni doimo chekli qadamlardan iborat qilib bo‘laklash imkoniyati mavjudligida. Ya’ni uni chekli sondagi oddiy ko‘rsatmalar ketma-ketligi shaklida ifodalash mumkin. Agar kuzatilayotgan jarayonni chekli qadamlardan iborat qilib qo‘llay olmasak, uni algoritm deb bo‘lmaydi.

Tushunarlilik. Biz kundalik hayotimizda berilgan algoritmlar bilan ishlayotgan elektron soatlar, mashinalar, dastgohlar, kompyuterlar, turli avtomatik va mexanik qurilmalarni kuzatamiz.

Ijrochiga tavsiya etilayotgan ko‘rsatmalar, uning uchun tushunarli mazmunda bo‘lishi shart, aks holda ijrochi oddiygina amalni ham bajara olmaydi. Undan tashqari, ijrochi har qanday amalni bajara olmasligi ham mumkin.

Har bir ijrochining bajarishi mumkin bo‘lgan ko‘rsatmalar yoki buyruqlar majmuasi mavjud, u ijrochining ko‘rsatmalar tizimi (sistemasi) deyiladi. Demak, ijrochi uchun berilayotgan har bir ko‘rsatma ijrochining ko‘rsatmalar tizimiga mansub bo‘lishi lozim.

Ko‘rsatmalarni ijrochining ko‘rsatmalar tizimiga tegishli bo‘ladigan qilib ifodalay bilishimiz muhim ahamiyatga ega. Masalan, quyi sinfning a’lochi o‘quvchisi "son kvadratga oshirilsin" degan ko‘rsatmani tushunmasligi natijasida bajara olmaydi, lekin "son o‘zini o‘ziga ko‘paytirilsin" shaklidagi ko‘rsatmani bemalol bajaradi, chunki u ko‘rsatma mazmunidan ko‘paytirish amalini bajarish kerakligini anglaydi.

Aniqlik. Ijrochiga berilayotgan ko‘rsatmalar aniq mazmunda bo‘lishi zarur. Chunki ko‘rsatmadagi noaniqliklar mo‘ljaldagi

maqsadga erishishga olib kelmaydi. Inson uchun tushunarli bo‘lgan "3-4 marta silkitilsin", "5-10 daqiqa qizdirilsin", "1-2 qoshiq solinsin", "tenglamalardan biri yechilsin" kabi noaniq ko‘rsatmalar robot yoki kompyuterni qiyin ahvolga solib qo‘yadi.

Bundan tashqari, ko‘rsatmalarning qaysi ketma-ketlikda bajarilishi ham muhim ahamiyatga ega. Demak, ko‘rsatmalar aniq berilishi va faqat algoritmda ko‘rsatilgan tartibda bajarilishi shart ekan.

Ommaviylik. Har bir algoritm mazmuniga ko‘ra bir turdagi masalalarning barchasi uchun ham o‘rinli bo‘lishi kerak. Ya’ni masaladagi boshlang‘ich ma’lumotlar qanday bo‘lishidan qat’iy nazar algoritm shu xildagi har qanday masalani yechishga yaroqli bo‘lishi kerak. Masalan, ikki oddiy kasrning umumiy maxrajini topish algoritmi, kasrlarni turlicha o‘zgartirib bersangiz ham ularning umumiy maxrajlarini aniqlab beraveradi. Yoki uchburchakning yuzini topish algoritmi, uchburchakning qanday bo‘lishidan qat’iy nazar, uning yuzini hisoblab beraveradi.

Natijaviylik. Har bir algoritm chekli sondagi qadamlardan so‘ng albatta natija berishi shart. Bajariladigan amallar ko‘p bo‘lsa ham baribir natijaga olib kelishi kerak. Chekli qadamdan so‘ng qo‘yilgan masala yechimga ega emasligini aniqlash ham natija hisoblanadi. Agar ko‘rilayotgan jarayon cheksiz davom etib natija bermasa, uni algoritm deb atay olmaymiz.

Algoritmning tasvirlash usullari .Yuqorida ko‘rilgan *misol*larda odatda biz masalani yechish algoritmini so‘zlar va matematik formulalar orqali ifodaladik. Lekin algoritm boshqa ko‘rinishlarda ham berilishi mumkin. Biz endi algoritmlarning eng ko‘p uchraydigan turlari bilan tanishamiz.

1.Algoritmning so‘zlar orqali ifodalanishi. Bu usulda ijrochi uchun beriladigan har bir ko‘rsatma jumlalar, so‘zlar orqali buyruq shaklida beriladi.

2. Algoritmning formulalar bilan berilish usulidan matematika, fizika, kimyo kabi aniq fanlardagi formulalarni o‘rganishda foydalaniladi. Bu usulni ba’zan analitik ifodalash deyiladi.

3. **Algoritmlarning grafik shaklida tasvirlanishida** algoritmlar maxsus geometrik figuralar yordamida tasvirlanadi va bu grafik ko‘rinishi blok-sxema deyiladi.

4. **Algoritmning jadval ko‘rinishda berilishi**. Algoritmning bu tarzda tasvirlanishdan ham ko‘p foydalanamiz. Masalan, maktabda

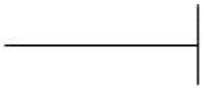
qo‘llanib kelinayotgan to‘rt xonali matematik jadvallar yoki turli xil lotereyalar jadvallari. Funksiyalarning grafiklarini chizishda ham algoritmlarning qiymatlari jadvali ko‘rinishlaridan foydalanamiz. Bu kabi jadvallardan foydalanish algoritmlari sodda bo‘lgan tufayli ularni o‘zlashtirib olish oson.

Yuqorida ko‘rilgan algoritmlarning tasvirlash usullarining asosiy maqsadi, qo‘yilgan masalani yechish uchun zarur bo‘lgan amallar ketma-ketligining eng qulay holatini aniqlash va shu bilan odam tomonidan dastur yozishni yanada osonlashtirishdan iborat. Aslida dastur ham algoritmning boshqa bir ko‘rinishi bo‘lib, u insonning kompyuter bilan muloqotini qulayroq amalga oshirish uchun mo‘ljallangan.

Blok-sxemalarni tuzishda foydalaniladigan asosiy sodda geometrik figuralardan iborat bo‘lib, u 1.1 jadvalda keltirilgan.

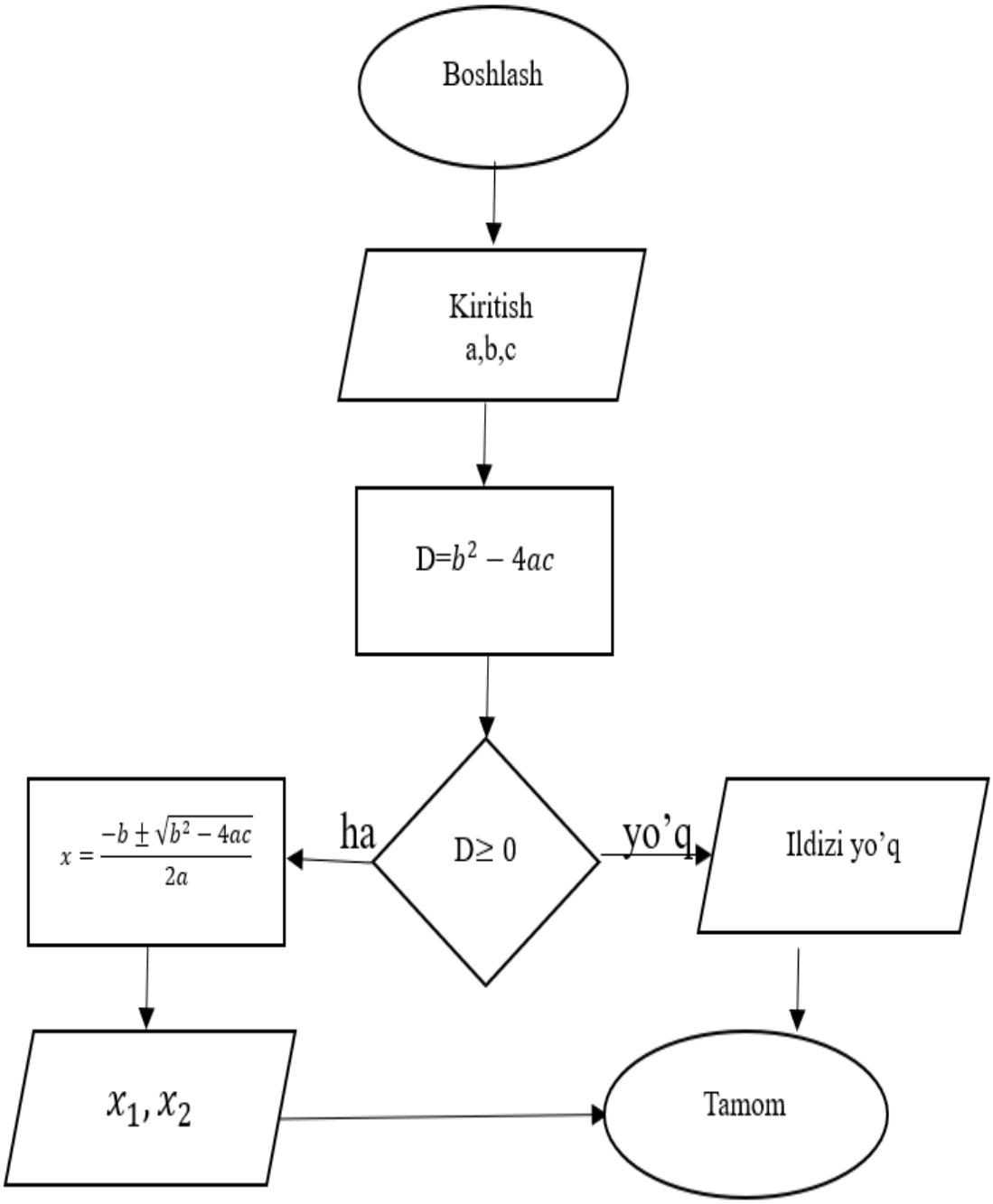
1.1.-jadval. Blok sxema ko‘rinishida ifodalaniladigan algoritmlarda ishlatiladigan figuralar va ularning vazifasi.

Nomi	Belgilanishi	Bajaradigan vazifasi
Jarayon		Bir yoki bir nechta amallarni bajarilishi natijasida ma’lumotlarning o‘zgarishi
Qaror		Biror shartga bog‘lik ravishda algoritmning bajarilish yo‘nalishini tanlash
Shakl o‘zgartirish		Dasturni o‘zgartiruvchi buyruq yoki buyruqlar turkumini o‘zgartirish amalini bajarish

Avval aniqlangan jarayon		Oldindan ishlab chiqilgan dastur yoki algoritmdan foydalanish
Kiritish Chiqarish		Axborotlarni qayta ishlash mumkin bo‘lgan shaklga o‘tkazish yoki olingan natijani tasvirlash
Displey		Kompyuterga ulangan monitordan axborotlarni kiritish yoki chiqarish
Xujjat		Axborotlarni qog’ozga chiqarish yoki qog’ozdan kiritish
Axborotlar oqimi chizig‘i		Bloklar orasidagi bog‘lanishlarni tasvirlash
Bog‘lagich		Uzilib qolgan axborot oqimlarini ulash belgisi
Boshlash Tugatish		Axborotni qayta ishlashni boshlash, vaqtincha yoki butunlay to‘xtatish
Izoh		Bloklarga tegishli turli xildagi tushuntirishlar

Blok-sxemalar bilan ishlashni yaxshilab o‘zlashtirib olish zarur, chunki bu usul algoritmlarni ifodalashning qulay vositalaridan biri bo‘lib, dastur tuzishni osonlashtiradi, dasturlash qobiliyatini mustahkamlaydi. Algoritmik tillarda blok — sxemaning asosiy strukturalariga maxsus operatorlar mos keladi. Shuni aytish kerakni, blok-sxemalardagi yozuvlar odatdagi yozuvlardan katta farq qilmaydi.

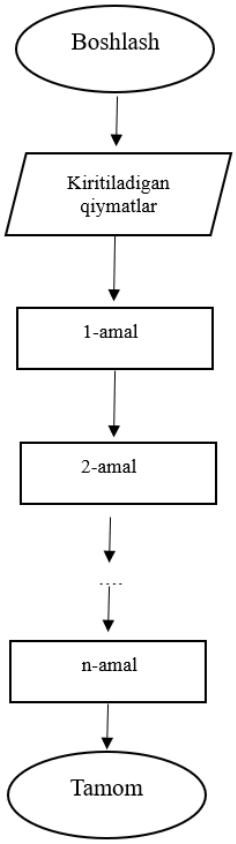
Misol sifatida $ax^2+bx+c=0$ kvadrat tenglamani yechish algoritmining blok-sxemasi quyida keltirilgan.



1.1-rasm. Kvadrat tenglamani yechish algoritmi

Chiziqli algoritmlar. Har qanday murakkab algoritmni ham uchta asosiy struktura yordamida tasvirlash mumkin. Bular ketma-ketlik, ayri va takrorlash strukturalaridir. Bu strukturalar asosida chiziqli, tarmoqlanuvchi va takrorlanuvchi hisoblash jarayonlarining algoritmlarini tuzish mumkin. Umuman olganda, algoritmlarni shartli ravishda quyidagi turlarga ajratish mumkin:

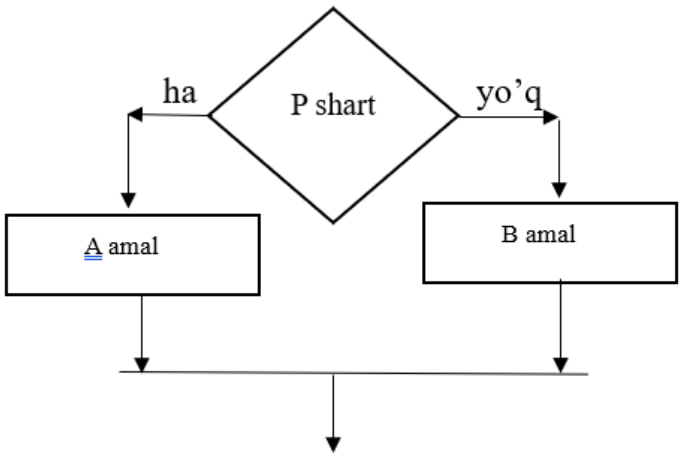
- chiziqli algoritmlar;
- tarmoqlanuvchi algoritmlar;
- takrorlanuvchi yoki siklik algoritmlar;
- ichma-ich joylashgan siklik algoritmlar;
- rekurrent algoritmlar;
- takrorlanishlar soni oldindan no‘malum algoritmlar;
- ketma-ket yaqinlashuvchi algoritmlar.



1.2 rasm. Chiziqli algoritmlar blok — sxemasining umumiy strukturasi

Faqat ketma-ket bajariladigan amallardan tashkil topgan algoritmlarga-chiziqli algoritmlar deyiladi. Bunday algoritmni ifodalash uchun ketma-ketlik strukturasi ishlatiladi. Strukturada bajariladigan amal mos keluvchi shakl bilan ko‘rsatiladi. Chiziqli algoritmlar blok-sxemasining umumiy strukturasini quyidagi ko‘rinishda ifodalash mumkin:

Tarmoqlanuvchi algoritmlar. Agar hisoblash jarayoni biror bir berilgan shartning bajarilishiga qarab turli tarmoqlar bo‘yicha davom ettirilsa va hisoblash jarayonida har bir tarmoq faqat bir marta bajarilsa, bunday hisoblash jarayonlariga tarmoqlanuvchi algoritmlar deyiladi. Tarmoqlanuvchi algoritmlar uchun ayri strukturasi ishlatiladi. Tarmoqlanuvchi strukturasi berilgan shartning bajarilishiga qarab ko‘rsatilgan tarmoqdan faqat bittasining bajarilishini ta’minlaydi.

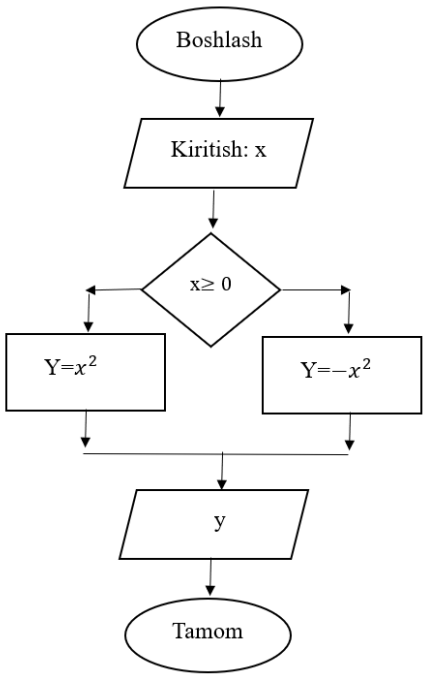


1.3-rasm. Tarmoqlanishning umumiy ko‘rinishi

Berilgan shart romb orqali ifodalanadi, P-berilgan shart. Agar shart bajarilsa, "ha" tarmoq bo‘yicha a amal, shart bajarilmasa "yo‘q" tarmoq bo‘yicha b amal bajariladi.

Tarmoqlanuvchi algoritmga tipik *misol* sifatida quyidagi sodda *misol*ni qaraylik.

1- Misol: $Y = \begin{cases} x^2, agar\ x \geq 0 \\ -x^2, agar\ x < 0 \end{cases}$

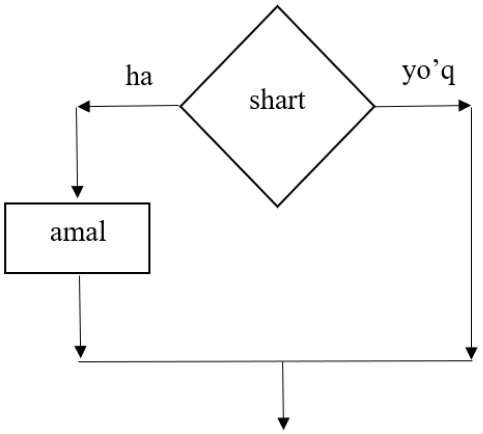


1.4-rasm.Interval ko‘rinishidagi funksiya qiymatini hisoblash algoritmi

Berilgan x ning qiymatiga bog‘lik holda, agar u musbat bo‘lsa «ha» tarmoq bo‘yicha $y=x^2$ funksiyaning qiymati, aks holda $y=-x^2$ funksiyaning qiymati hisoblanadi. Ko‘pgina masalalarni yechishda, shart asosida tarmoqlanuvchi algoritmlarning ikkita tarmog‘idan bittasining, ya’ni yoki «ha» yoki «yo‘q» ning bajarilishi yetarli bo‘ladi. Bu holat tarmoqlanuvchi algoritmning xususiy holi sifatida

aylanish strukturasi deb atash mumkin. Aylanish strukturasi quyidagi ko‘rinishga ega:

Takrorlanuvchi algoritmlar .Agar biror masalani yechish uchun tuzilgan zarur bo‘lgan amallar ketma-ketligining ma’lum bir qismi biror parametrga bog‘liq ko‘p marta qayta bajarilsa, bunday algoritm takrorlanuvchi algoritm yoki siklik algoritmlar deyiladi. Takrorlanuvchi algoritmlarga tipik *misol* sifatida odatda qatorlarning yig‘indisi yoki ko‘paytmasini hisoblash jarayonlarini qarash mumkin. Quyidagi yig‘indini hisoblash algoritmini tuzamiz:

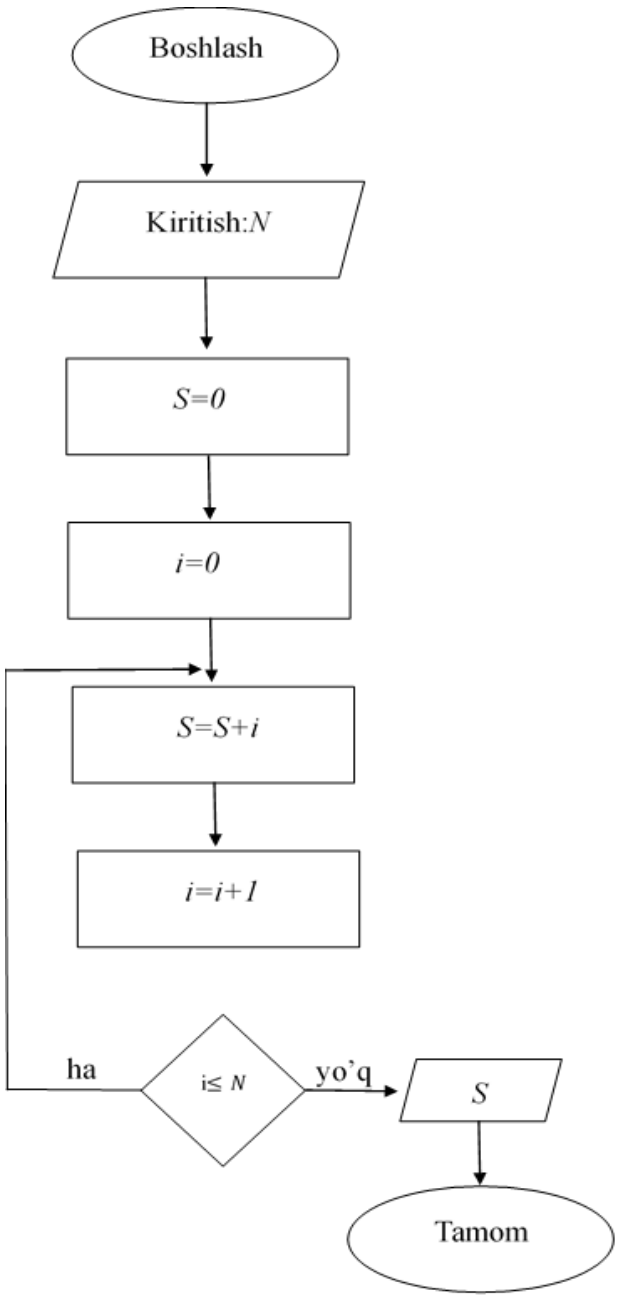


1.5-rasm. Aylanish strukturasining umumiy ko‘rinishi

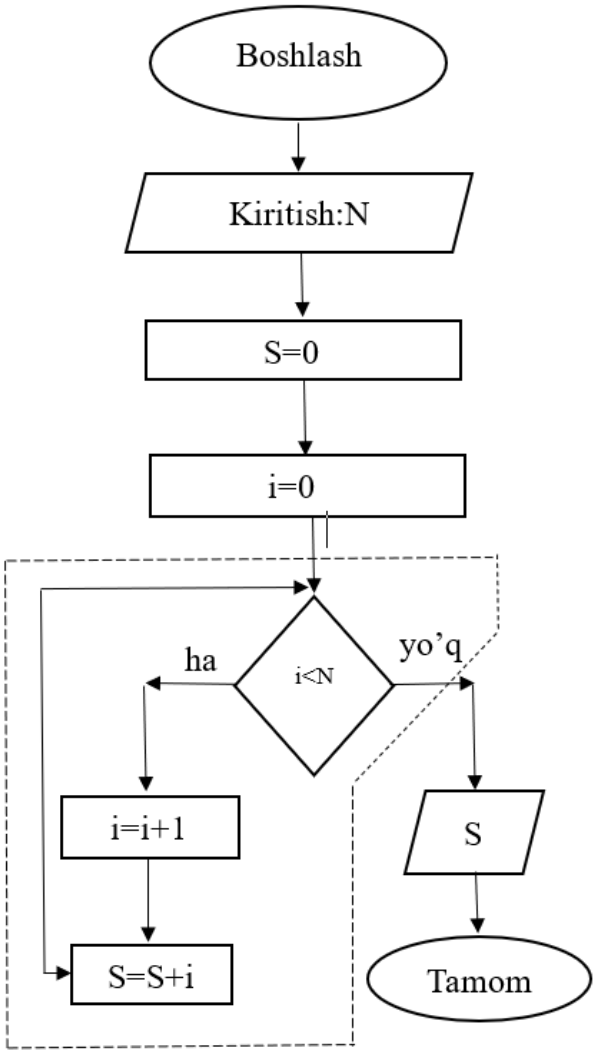
$$S = 1 + 2 + 3 + ... + N = \sum_{i=1}^{N} i$$

Bu yig‘indini hisoblash uchun i=0 da S=0 deb olamiz va $i=i+1$ da $S=S+i$ ni hisoblaymiz. Bu yerda birinchi va ikkinchi qadamlar uchun yig‘indi hisoblanadi va keyingi qadamda i parametr yana bittaga ortiriladi va navbatdagi raqam avvalgi yig‘indi S ning ustiga qo‘shiladi va bu jarayon shu tartibda to $i<N$ sharti bajarilmaguncha davom ettiriladi va natijada izlangan yig‘indiga ega bo‘lamiz. Bu fikrlarni quyidagi algoritm sifatida ifodalash mumkin:
N –berilgan bo‘lsin,
$i=0$ berilsin,
$S=0$ berilsin,
$i=i+1$ hisoblansin,
$S=S+i$ hisoblansin,

i<*N* tekshirilsin va bu shart bajarilsa, 4-satrga qaytilsin, aks holda keyingi qatorga o‘tilsin, *S* ning qiymati chop etilsin.



1.6-rasm.1 dan *n* gacha bo‘lgan sonlar yig‘indisini hisoblash algoritmi

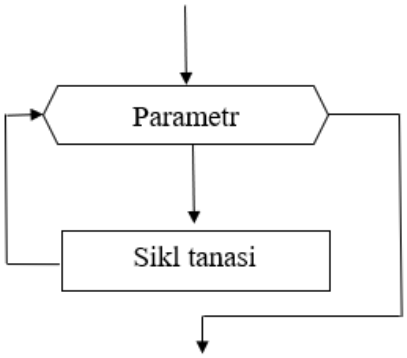


1.7-rasm. 1 dan *n* gacha bo‘lgan sonlar yig‘indisini hisoblash algoritmi

Yuqorida keltirilgan algoritm va blok sxemadan (1.6-rasm) ko‘rinib turibdiki amallar ketma-ketligining ma’lum qismi parametr *i* ga nisbatan N marta takrorlanadi.

Yuqorida ko‘rilgan yig‘indi blok sxemalaridagi takrorlanuvchi qismlariga (aylana ichiga olingan) quyidagi sharti keyin berilgan siklik struktura mos kelishini ko‘rish mumkin. Yuqoridagi blok sxemalarda shartni oldin tekshiriladigan holatda chizish mumkin edi. Masalan, yig‘indining algoritmini qaraylik. Bu blok sxemaning takrorlanuvchi qismiga quyidagi, sharti oldin berilgan siklik strukturaning mos kelishini ko‘rish mumkin.

Blok sxemalarining takrorlanuvchi qismlarini, quyidagi parametrli takrorlash strukturasi ko‘rinishida ham ifodalash mumkin.



1.8-rasm. Parametrli takrorlash operatorining umumiy ko‘rinishi

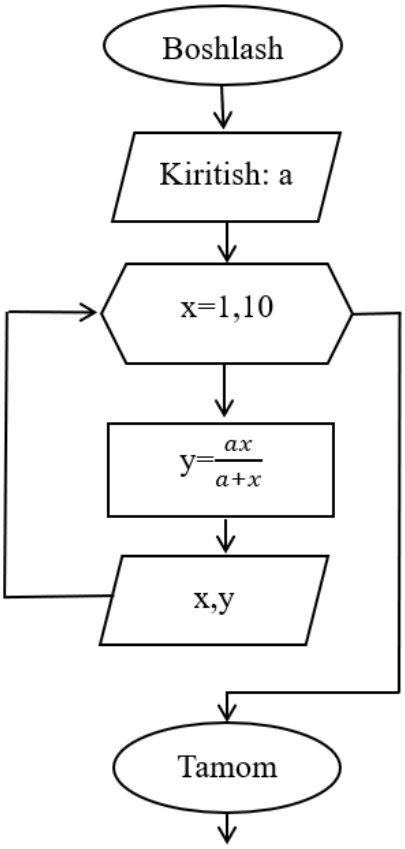
Parametrli takrorlash operatoriga misol sifatida berilgan x=1,2,3,.....10 larda $y = \frac{ax}{a+x}$ funksiyasining qiymatlarini hisoblash blok sxemasini qarash mumkin (1.9-rasm.).

Ichma-ich joylashgan siklik algoritmlar. Ba’zan, takrorlanuvchi algoritmlar bir nechta parametrlarga bog‘liq bo‘ladi. Odatda bunday algoritmlarni ichma-ich joylashgan algortmlar deb ataladi.

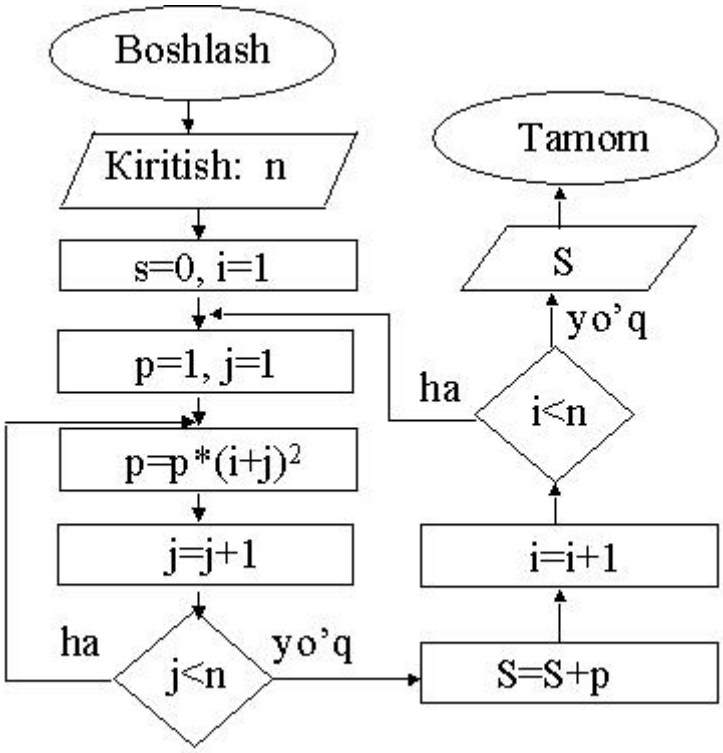
Misol sifati berilgan nxm o‘lchovli a_{ij} –matritsa elementlarining yig‘indisini hisoblash masalasini qaraylik.

$$S = \sum_{i=1}^{n} \prod_{j=1}^{n} (i+j)^2$$

Bu yig‘indi hisoblash uchun, *i* ning har bir qiymatida *j* bo‘yicha ko‘paytmani hisoblab, avval yig‘indi ustiga ketma-ket qo‘shib borish kerak bo‘ladi. Bu jarayon quyidagi blok–sxemada aks ettirilgan. Bu yerda *i*-tashqi sikl — yig‘indi uchun, *j*-esa ichki sikl-ko‘paytmani hosil qilish uchun foydalanilgan.



1.19-rasm. Parametrli takrorlash operatoriga doir algoritm



1.10-rasm. Ichma-ich joylashgan siklik algoritmga doir blok-sxema

2-mavzu. C++ dasturlash tilining tuzilishi va shakli.

C++ dasturlash tili to‘g‘risidagi asosiy tushunchalar:

- C++ bu yuqori unumli ilovalar yaratish uchun ishlatilishi mumkin bo‘lgan o‘zaro faoliyat platformalar tili;
- C++ Bjarne Stroustrup tomonidan C tilining kengaytmasi sifatida ishlab chiqilgan;
- C++ dasturchilarga tizim resurslari va xotira ustidan yuqori darajadagi nazoratni beradi;
- Til 2011,2014,2017 va 2020 yillarda C++1, C++14,C++17,C++20 ga 4 marta yangilandi.

C++ dasturlash tili afzalliklari:

- ✓ C++ dunyodagi eng mashhur dasturlash tillaridan biridir;
- ✓ C++ tilini hozirgi operatsion tizimlarda, foydalanuvchi grafik interfeyslarida va o‘rnatilgan tizimlarda topish mumkin;
- ✓ C++ bu ob’yektga yo‘naltirilgan dasturlash tili bo‘lib, u dasturlarga aniq tuzilma beradi, kodni qayta ishlatishga imkon beradi va ishlab chiqish xarajatlarini kamaytiradi;
- ✓ C++ portativ hisoblanadi va bir nechta platformalarga moslasha oladigan ilovalarni ishlab chiqish uchun ishlatilishi mumkin;
- ✓ C++ bu qiziqarli va o‘rganish oson;
- ✓ C++ tili C, C# va Java tillariga yaqin bo‘lgani uchun dasturchilar uchun C++ tiliga yoki aksincha o‘tishni osonlashtiradi.

C va C++ versiyasi o‘rtasidagi farqlar:

- C++ tili C tilining kengaytmasi sifatida ishlab chiqilgan va ikkala til ham deyarli bir xil sintaksisga ega;
- C va C++ o‘rtasidagi asosiy farq shundaki , C++ sinflar va ob’yektlarni qo‘llab quvvatlaydi, C esa qo‘llab quvvatlamaydi.

C++ dasturlash tili sintaksisi.

C++ sintaksisini yaxshiroq tushunishimiz uchun quyidagi kodni yozamiz va alohida qatorlarga ajratib tavsif beramiz:

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
cout<<"Hello woorld!";
return 0;
}
	Hello world!

Misol tushuntirilishi:

1-**qator:**(5-qatorda foydalanilgan) kabi kiritish va chiqarish ob'yektlari bilan ishlashga imkon beruvchi sarlavha fayllari kutubxonasi #include<iostream> . Sarlavha fayllari C++ dasturlariga funksionallik qo'shadi.

2-**qator:** using namespace std; standart kutubxonadagi ob'yektlar va o'zgaruvchilar uchun nomlardan foydalanishimiz mumkinligini anglatadi.

3-**qator:** Bo'sh qator. C++ bo'sh qatorga e'tibor bermaydi. Lekin biz undan kodni o'qilishi mumkin bo'lishi uchun foydalanamiz.

4-qator:C++ dasturida har doim ishlatiladigan yana bir narsa int main(). Bu funksiya deyiladi. Figurali qavslar ichidagi har qanday kod {} ichida bajariladi.

5-qator: cout matnni chiqarish/chop etish uchun *kiritish operatori()* bilan birgalikda foydalaniladigan **obyekt**. Bizning misolimizda"Hello world!" deb chiqadi.

Yodda saqlash uchun: Har bir C++ bayonoti nuqtali vergul bilan tugaydi. Kompilyator bo'shliqlarga e'tibor bermaydi. Biroq bir nechta satr kodni o'qishni yanada qulay qiladi.

6-**qator:** return 0 asosiy funksiyani tugatadi.

7-qator: } asosiy funksiyani amalda tugatish uchun figurali qavs qo'yishni unutmang!

C++ algoritmik tilining alifbosi quyidagilardan iborat:

- Katta va kichik lotin harflari;
- 0 dan 9 gacha raqamlar;
- Maxsus belgilar (+,-,*,/,=,[,],<,>,{,}….) ni o‘z ichiga oladi.

C++ tilida so‘z deb bir nechta belgilar ketma ketligi tushuniladi.xizmatchi so‘z deb c++ tilidagi standart nom tushuniladi.

C++ tilida ma’lumotlarning elementlari bo‘lib **o‘zgaruvchilar**, **o‘zgarmaslar**, **izohlar** xizmat qiladi.

O‘zgaruvchilar:

Xotiraning nomlangan qismi bo‘lib, o‘zida ma’lum bir toifadagi qiymatlarni saqlaydi. O‘zgaruvchining nomi va qiymati bo‘ladi. O‘zgaruvchining nomi orqali qiymat saqlanayotgan xotira qismiga murojaat qilinadi. Programma ishlashi jarayonida o‘zgaruvchining qiymatini o‘zgartirish mumkin. Har qanday o‘zgaruvchini ishlatishdan oldin uni e’lon qilish kerak bo‘ladi.

O‘zgarmaslar:

Hisoblash jarayonida qiymatini o‘zgartirmaydigan kattaliklarga aytiladi.

Izohlar:

Dasturning ma’lum qismini tavsiflash uchun ishlatiladi va bu qatorda hech qanday amal bajarilmaydi, ya’ni programmaning biror qismini yaxshiroq tushunish uchun xizmat qiladi.izoh “/*” va “*/” simvollari orasida beriladi.

Operator tilning yakunlangan jumlasi hisoblanadi va ma’lumotlar tahlilining bosqichini ifodalaydi. Operatorlar nuqtali vergul “;” bilan ajratiladi. C++ da operatorlar programmada keltirilgan ketma-ketlikda bajariladi.

Identifikator programmist tomonidan programma elementlari (funksiya,o‘zgaruvchilar,o‘zgarmaslar….) uchun ixtiyoriy tanlangan nom.

Identifikator tanlanganda quyidagilarga amal berish kerak:

- ✓ Identifikator lotin harflaridan boshlanishi shart;
- ✓ Ikkinchi simvoldan boshlab raqamlardan foydalanish mumkin;
- ✓ C++ da katta kichik harflar farq qiladi. Ya’ni quyidagilarning har biri alohida identifikator hisoblanadi: KATTA,katta,Katta,KaTta,…

✓ Bo‘sh joy (probel) c++ da so‘zlarni ajratish uchun ishlatiladi. Shuning uchun identifikatorda bo‘sh joydan foydalanib bo‘lmaydi;

✓ Xizmatchi (int,float,for,while) kabi Identifikator sifatida ishlatib bo‘lmaydi.

C++ da programma funksiya yoki funksiyalardan tashkil topadi. Agar programma bir nechta funksiyadan iborat bo‘lsa bir funksiyaning nomi ***main*** bo‘lishi shart. Programma aynan ***main*** funksiyasining birinchi operatoridan boshlab bajariladi.

Qoida bo‘yicha funksiya qandaydir bir qiymatni hisoblash uchun ishlatiladi. Shuning uchun funksiya nomi oldidan funksiya qaytaradigan qiymat toifasi yoziladi. Agar funksiya hech qanday qiymat qaytarmaydigan bo‘lsa, ***void*** toifasi yoziladi. Agar funksiya qaytaradigan qiymat toifasi yozilmagan bo‘lsa, int(butun toifali) qiymat qaytaradi deb qabul qilinadi.

Klaviaturadan ma’lumotlarni kiritish va ekranga chiqarish uchun <**iostream**> sarlavha faylidan foydalaniladi. Bu satr klaviatura orqali ma’lumot kirituvchi va ekranga nimadir chiqaruvchi har qanday dasturda bo‘lishi shart. Ba’zi eski kompilyatorlarda <**iostream.h**> yozilishi mumkin.

Dasturda ishlatiladigan barcha o‘zgaruvchilarni qaysi toifaga tegishli ekanligini e’lon qilish kerak. Ma’lumotlarni e’lon qilishning umumiy ko‘rinishi quyidagicha:

toifa_nomi o‘zgaruvchi;

Agar bir nechta o‘zgaruvchi bir toifaga mansub bo‘lsa, ularni vergul bilan ajratish mumkin.

Dasturda ma’lumot kiritish va ekranga chiqarish uchun **#include<iostream>** ni dasturga qo‘shish shart.

Ma’lumotlarni kiritish **std::cin>>** , ma’lumotlarni chiqarish **std::cout<<** operatori orqali amalga oshiriladi.

Quyida a va b sonlarining yig‘indisini topadigan dastur tuzamiz.

```
#include<iostream>
using namespace std;
int main(){
int a,b,c;
cout<<”a=”; cin>>a;
cout<<”b=”;
cin>>b;
c=a+b;
cout<<c<<endl;
return 0;
}
```

3-mavzu. O‘zgaruvchilar va ifodalar.

O‘zgaruvchi—*Xotiraning nomlangan qismi bo‘lib, o‘zida ma’lum bir toifadagi qiymatlarni saqlaydi. O‘zgaruvchining nomi va qiymatlari bo‘ladi. O‘zgaruvchining nomi orqali qiymat saqlanayotgan xotira qismiga murojaat qilinadi. Programma ishlashi jarayonida o‘zgaruvchining qiymatini o‘zgartirish mumkin. Har qanday o‘zgaruvchini ishlatishdan oldin, uni e’lon qilish lozim.*

- **int**— 123 yoki -123 kabi butun sonlarni (butun sonlar) saqlaydi.
- **double** - 19.99 yoki -19.99 kabi o‘nli raqamlar raqamlarini saqlaydi
- **char**- "a" yoki "B" kabi bitta belgilarni saqlaydi. **Char** qiymatlari bitta tirnoqli qavslar bilan o‘ralgan.
- **string**- "Hello World" kabi matnlarni saqlaydi. Qator qiymatlari qo‘shtirnoq bilan o‘ralgan
- **bool** - qiymatlarni ikki holat bilan saqlaydi: rost yoki yolg‘on, **True** yoki **False** (**1** yoki **0**)

O‘zgaruvchi qisqa nomga ega bo‘lishi mumkin (masalan, x va y) yoki ko‘proq tavsiflovchi nom (yosh, cpp, master va h.k). **C++** o‘zgaruvchilar uchun qoidalar:

- O‘zgaruvchilar nomi raqam bilan boshlanmasligi kerak;
- O‘zgaruvchi **[a-Z]** harf bilan boshlanib raqamlar ham aralashtirish mumkin *(misol: _ali123, son90 va h.k)*
- O‘zgaruvchi nomida katta va kichik harflar ahamiyatga ega bo‘lib misol uchun **Aka** va **aka** ikki xil o‘zgaruvchi nomi hisoblanadi! (Aka, aka, AkA, aKa)

Dastur biror bir aniqlangan ketma — ketlikda bajariluvchi komandalar to‘plamidan iborat. C++ tilida ifodalar biror bir hisoblash natijasini qaytaruvchi boshqa ifodalar ketma-ketligini boshqaradi yoki hech nima qilmaydi (nol ifodalar).

C++ tilida barcha ifodalar nuqtali vergul bilan yakunlanadi. Ifodaga misol qilib o‘zlashtirish amalini olish mumkin.

x=a+b;

Algebradan farqli ravishda bu ifoda x o’zgaruvchisi a+b ga teng ekanligini anglatmaydi. Bu ifodani quyidagicha tushunish kerak:

a va b o‘zgaruvchilarni qiymatlarini yig‘ib natijasini x o‘zgaruvchiga beramiz yoki x o‘zgaruvchiga a+b qiymatni o‘zlashtiramiz. Bu ifoda birdaniga ikkita amalni bajaradi, yig‘indini

hisoblaydi va natijani o‘zlashtiradi. Ifodadan so‘ng nuqtali vergul qo‘yiladi. (=) operatori o‘zidan chap tomondagi operandga o‘ng tomondagi operandlar ustida bajarilgan amallar natijasini o‘zlashtiradi.

Bo‘sh joy (probel) belgisi.

Bo‘sh joy belgilariga nafaqat probel, balki yangi satrga o‘tish va tabulyasiya belgilari ham kiradi. Yuqorida keltirilgan ifodani quyidagicha ham yozish mumkin:

x= a+ b;

Bu variantda keltirilgan ifoda ko‘rimsiz va tushunarsiz bo‘lsa ham to‘g‘ridir. Bo‘sh joy belgilari dasturning o‘qilishini ta’minlaydi. Bloklar va kompleks ifodalar.

Ba’zan dastur tushunarli bo‘lishi uchun o‘zaro mantiqiy bog‘langan ifodalarni blok deb ataluvchi komplekslarga birlashtirish qulay hisoblanadi. Blok ochiluvchi figurali qavs ({) bilan boshlanadi va yopiluvchi figurali qavs (}) bilan tugaydi. Blok ochilganda va yopilganda nuqtali vergul qo‘yilmaydi.

```
{   temp= a;
  a = b;
  b = temp; }
```

Bu blok xuddi bir ifodadek bajariladi, u a va v o‘zgaruvchilar qiymatlarini almashtiradi.

Ishlatishdan oldin har qanday o‘zgaruvchini aniqlash kerak. O‘zgaruvchini aniqlash sintaksisi quyidagicha:

o‘zgaruvchining _turi o‘zgaruvchining_nomi;

O‘zgaruvchilarni nomlash.

O‘zgaruvchining nomi alfavit-raqamli belgilar ketma-ketligi va pastki chiziq belgisi _ bilan yaratilishi mumkin. Bunday holda, o‘zgaruvchining nomi alifbo belgisi yoki pastki chiziq bilan boshlanishi kerak.

Bundan tashqari, C++ kalit so‘zlarini o‘zgaruvchining nomi sifatida ishlatish mumkin emas, ya’ni *for* yoki *if* kalit so‘zlarini ishlatish mumkin emas. Ammo bunday so‘zlar unchalik ko‘p emas va C++ tilini o‘rganish jarayonida siz kalit so‘zlarni ishlatish jarayonida tezda yodlab olasiz qaysi so‘zlar muhimligini bilib olasiz.

Shuni ta'kidlash kerakki, quyidagi nomlar tavsiya etilmaydi:

- ikkita pastki chiziq bilan boshlanadigan ismlar (Masalan __Nom);
- pastki chiziq bilan boshlanadigan ismlar, so‘ngra bosh alifbo belgisi (Masalan, _A12);
- pastki chiziq bilan boshlanadi global domendagi nomlar (main funktsiyadan tashqari);

Aslini olganda, bunday nomlar bilan kompilyator tomonidan ishlab chiqarilgan yoki C++plaginli standart modullarida aniqlangan nomlar (masalan, o‘zgaruvchan nomlar) bilan to‘qnashishi ehtimoli ortadi. Shuning uchun, ba'zi dasturchilar o‘zgaruvchi nomini pastki chiziq bilan boshlashni umuman tavsiya etmaydi.

Initsializatsiya.

O‘zgaruvchini aniqlagandan so‘ng, ba'zi bir qiymat berilishi mumkin. O‘zgaruvchiga boshlang‘ich qiymatni berish *initsializatsiya* deb ataladi. C++ da initsializatsiyaning uch turi mavjud:

- O‘zlashtirish notatsiyasi (assignment notation);
- Funksional notatsiya (functional notation)
- Figurali qavslarda initsializatsiya (braced initialization)

Endi yuqorida keltirilgan initsializatsiya turlari haqida batafsil tanishib chiqamiz.

O‘zlashtirish notatsiyasining mohiyati-o‘zgaruvchiga initsializatsiy operatoridan ("teng" yoki =belgisi) foydalanib, biz ba'zi qiymatlarni o‘яlashtiramiz:

```
int age;
age = 20;
```

Bu erda o‘zgaruvchining qiymati sifatida 20 raqami berildi. 20, 123.456 (kasr son), "A" yoki "salom" kabi raqamlar, belgilar, satrlar kabi har qanday turdagi doimiy qiymatlar literal deb ataladi. Ya'ni, bu holda, o‘zgaruvchiga butun son literal 20 tayinlanadi.

Funksional notatsiya. Funktsional belgilar bilan, o‘zgaruvchi nomidan keyin uning qiymati qavs ichida ko‘rsatiladi:

```
int age (38);   // functional notation
```

Bunday holda, o‘zgaruvchi 38 qiymatiga ega bo‘ladi.

Quyida keltirilgan uch holatda ham o‘zgaruvchiga berilgan qiymat murakkab baholash ifodasini ko‘rsatishi mumkin. Masalan:

```
int age1 {22 + 5};
int age2 (22 + 5);
```

```
int age3 = 22 + 5;
```

Figurali qavslarda initsializatsiya. Figurali qavsli initsializatsiyani ishga tushirishda, o‘zgaruvchining nomidan keyin uning qiymati figurali qavslar ichida ko‘rsatiladi:

```
int age {38};   // braced initialization
```

Yuqorida keltirilgan holatlarni takrorlash uchun ketma-ket bir vaqtning o‘zida bir nechta o‘zgaruvchilarni ishga tushirishingiz mumkin:

```
int age1 {22}, age2 (23), age3 = 24;
```

Ko‘pgina hollarda, initsializatsiyaning uchta varianti ham tengdir. Biroq, aniq toifadagi transformatsiya qo‘llanilganda, figurali qavslarda ishga tushirish biroz xavfsizroq bo‘ladi. Umuman olganda, o‘zgaruvchiga uning turiga mos keladigan qiymat berilishi kutiladi. Agar bunday bo‘lmasa, kompilyator tayinlangan qiymatni o‘zgaruvchi turiga aylantirishga harakat qiladi. Konstriktiv transformatsiya bir turdagi qiymatni cheklangan qiymat diapazoniga o‘zgartiradi. Shunday qilib, konvertatsiya ma'lumotlarning yo‘qolishiga olib kelishi mumkin. Quyidagi misolda buni ko‘rishimiz mumkin:

```
#include <iostream>

int main()
{
   int age1 (23.5);
   int age2 = 24.5;
   std::cout<<"Age1 = " << age1 << "\n";
   std::cout<<"Age2 = " << age2 << "\n";
}
```

Bu yerda int tipini, ya'ni butun sonni ifodalovchi age1 va age2 o‘zgaruvchilarga mos ravishda 23,5 va 24,5 kasr qiymati berildi. Ammo ko‘pchilik kompilyatorlarda, bu kod odatdagidek kompilyatsiya qilinadi va ishlaydi. Biz quyidagi natijani olamiz:

Age1 = 23
Age2 = 24

Endi figurali qavslar orqali ishga tushirishga misol keltiramiz:

```
#include <iostream>

int main()
{
   int age {22.5};
   std::cout<<"Age = " << age << "\n";
}
```

Bu erda butun sonni ifodalovchi Age o‘zgaruvchisiga 22.5 ham kasr qiymati beriladi. Biroq, endi kompilyatsiya paytida kompilyatorbizga xato haqida xabar beradi:

```
Compilation failed due to following error(s).
main.cpp: In function ‘int main()’:
main.cpp:5:18: error: narrowing conversion of ‘2.25e+1’ from ‘double’ to ‘int’ [-Wnarrowing]
    5 |     int age {22.5};
      |                  ^
```

Nol yordamida initsializatsiya qilish.

Figurali qavslar ichida initsializatsiya qilishda qiymatni o‘tkazib yuborishingiz mumkin:

int counter {};

yoki ushbu figurali qavsning ichida nol yozish orqali ham amalga oshirish mumkin:

int counter {0};

4-mavzu. C++ tilining amallari. Inkrement, decrement, sizeof, mantiqiy, razryadli, taqqoslash. Amallarning ustunliklari va bajarish yo‘nalishlari.

Amallar. Bajarilishi natijasida biror bir qiymat qaytaradigan barcha ifodalar C++ tilida amallar deyiladi. Amallar albatta biror bir qiymat qaytaradi. Masalan, 3+2 amali 5 qiymatni qaytaradi.

Operator — bu qandaydir amalni bajarish to‘g‘risida kompilyatorga uzatiladigan literaldir. Operatorlar operandlarga ta’sir qiladi. C++ da operandlar deb alohida literallar va butun ifodalar tushuniladi.

C++ tilida ikki ko‘rinishdagi operatorlar bor:

o‘zlashtirish operatorlari

matematik operatorlar

O‘zlashtirish operatori.

O‘zlashtirish operatori (=) o‘zidan chap tomonda turgan operand qiymatini tenglik belgisidan o‘ng tomondagilarni hisoblangan qiymatiga almashtiradi. Masalan,

x = a+b;

ifodasi x operandga a va v o‘zgaruvchilarni qiymatlarini qo‘shishdan hosil bo‘lgan natijani o‘zlashtiradi.

O‘zlashtirish operatoridan chapda joylashgan operand adresli operand yoki l–qiymat (inglizcha *left*-chap so‘zidan olingan) deyiladi. O‘zlashtirish operatoridan o‘ngda joylashgan operand operasion operand yoki r–qiymat deyiladi.

O‘zgarmaslar faqatgina r–qiymat bo‘lishi mumkin va hech qachon adresli operand bo‘la olmaydi, chunki dasturning bajarilishi jarayonida o‘zgarmas qiymatini o‘zgartirib bo‘lmaydi.

Matematik operatorlar.

C++ tilida 5 ta asosiy matematik operatorlar ko‘llaniladi: qo‘shish (+), ayirish (-), ko‘paytirish (*), butun songa bo‘lish (\) va modul bo‘yicha bo‘lish (%)(qoldiqni olish).

Ishorasiz butun sonlarni ayirishda, agarda natija manfiy son bo‘lsa g‘ayrioddiy natija beradi.

Listing 4.1. Ayirish natijasida butun sonni to‘lib qolishiga misol.

```
# include < iostream.h >
int main()
{
unsiegned int ayirma
```

```
unsisgned int kattaSon = 100;
unsigned int kichikSon = 50;
ayirma = kattaSon – kichikSon;
cout << “Ayirma“:<< ayirma<< “ ga teng\n”;
ayirma = kichikSon — kattaSon ;
cout << “Ayirma“:<< ayirma<< “ ga teng\n”;
endl;
return 0;
    }
```

NATIJA:

Ayirma: 50 ga teng

Ayirma: 4294967246 ga teng

Butun songa bo‘lish va qoldiqni olish operatorlari.

Butun songa bo‘lish odatdagi bo‘lishdan farq qiladi. Butun songa bo‘lishdan hosil bo‘lgan bo‘linmaning faqatgina butun qismi olinadi. Masalan, 21 sonini 4 ga bo‘lsak 5 soni va 1 qoldiq hosil bo‘ladi. 5 butun songa bo‘lishni qiymati, 1 esa qoldiqni olish qiymati hisoblanadi. Agar qiymat o’zlashtiruvchi o‘zgaruvchi butun son bo‘lsa, / amali bo‘linmaning butun qismini oladi. Qoldiq qismini olish uchun % belgisidan foydalaniladi.

Inkrement va dekrement.

Dasturlarda o‘zgaruvchiga 1 ni qo‘shish va ayirish amallari juda ko‘p hollarda uchraydi. C++ tilida qiymatni 1 ga oshirish inkrement, 1 ga kamaytirish esa dekrement deyiladi. Bu amallar uchun maxsus operatorlar mavjuddir.

Inkrement operatori (++) o‘zgaruvchi qiymatini 1 ga oshiradi, dekrement operatori (--) esa o‘zgaruvchi qiymatini 1 ga kamaytiradi. Masalan, s o‘zgaruvchisiga 1 qiymatni qo‘shmoqchi bo‘lsak quyidagi ifodani yozishimiz lozim.

C++ //s o‘zgaruvchi qiymatini 1 ga oshirdik.

Yuqoridagi ifodani quyidagicha yozishimiz mumkin edi.

```
s=s+1;
```

Bu ifoda o‘z navbatida quyidagi ifodaga teng kuchli:

```
s+=1;
```

Prefiks va postfiks.

Inkrement operatori ham, dekrement operatori ham ikki variantda ishlaydi: prefiksli va postfiksli. Perefiksli variantda ular o‘zgaruvchidan

oldin (++Age), postfiksli variantda esa o‘zgaruvchidan keyin (Age++) yoziladi.

Oddiy ifodalarda bu variantlarni qo‘llanishida farq katta emas, lekin bir o‘zgaruvchiga boshqa o‘zgaruvchining qiymatini o‘zlashtirishda ularning qo‘llanilishi boshqacha xarakterga ega. Perefeksli operator qiymat o‘zlashtirilguncha, postfiksli operator esa qiymat o‘zlashtirilgandan keyin bajariladi. Buni quyidagi listingdan ko‘rishimiz mumkin:

Listing 4.2. Prefiksli va postfiksli operatorlarni qo‘llanishi.

```
# include < iostream. h >
int main()
{
int myAge = 39;
int yourAge = 39;
cout << “Men” << MyAge <<”yoshdaman \n”;
cout << “Siz” << yourAge <<“yoshdasiz \n”;
myAge++ ; // postfiksli inkrement
++yourAge; // prefeksli inkrement
cout << “Bir yil o‘tdi …\ n”;
cout << “Men” << MyAge <<”yoshdaman \n”;
cout << “Siz” << yourAge <<“yoshdasiz \n”;
cout << “Yana bir yil o‘tdi \n”;
cout <<” Men”<< myAge++ <<“yoshdaman \n”;
cout <<”Siz”<<++yourAge<<“yoshdasiz \n”;
cout << “Ma’lumotlarni qaytadan”
cout << “chiqaraylik \n”;
cout <<” Men”<< myAge<<“yoshdaman \n”;
cout <<”Siz”<<yourAge<<“yoshdasiz \n”;
return 0;
}
```

NATIJA:
Men 39 yoshdaman
Siz 39 yoshdasiz
Bir yil o‘tdi …
Men 40 yoshdaman
Siz 40 yoshdasiz
Yana bir yil o‘tdi …

Men 40 yoshdaman
Siz 41 yoshdasiz
Ma’lumotlarni qaytadan chiqaraylik
Men 41 yoshdaman
Siz 41 yoshdasiz

Operatorlar prioriteti.

Murakkab ifodalarda qaysi amal birinchi navbatda bajariladi, qo‘shishmi yoki ko‘paytirishmi? Masalan:

```
x=5+3*8;
```

Agarda ifodada birinchi qo‘shish bajarilsa natija 64 ga, agarda ko‘paytirish birinchi bajarilsa natija 29 ga teng bo‘ladi.

Har bir operator prioritet qiymatiga ega. Ko‘paytirish qo‘shishga nisbatan yuqoriroq prioritetga ega. Shuning uchun bu ifoda qiymati 29 ga teng bo‘ladi.

Agarda ikkita matematik ifodaning prioriteti teng bo‘lsa, ular chapdan o‘ngga qarab ketma – ket bajariladi.

Demak, x=5+3+8*9+6*4 ifodada birinchi ko‘paytirish amallari chapdan o‘ngga qarab bajariladi 8*9=72 va 6*4=24. Keyin bu ifoda soddaroq ko‘rinish hosil qiladi: x=5+3+72+24

Endi qo‘shishni ham xuddi shunday chapdan unga qarab bajaramiz:

5+3=8; 8+72= 80; 80+24=104;

Lekin, barcha operatorlar ham bu tartibga amal qilmaydi. Masalan, o‘zlashtirish operatori o‘ngdan chapga qarab bajariladi.

Ichki qavslar

Murakkab ifodalarni tuzishda ichki qavslardan foydalaniladi. Masalan, sizga sekundlarning umumiy soni keyin esa barcha qaralayotgan odamlar soni, undan keyin esa ularning ko‘paytmasini hisoblash kerak bo‘lsin.

```
TotalPersonSeconds=((NumMinutesToThink+
NumMinutesToType)*60*(PeopleInTheOffice+ PeopleOnVocation ))
```

Bu ifoda quyidagicha bajariladi. Oldin NumMinutesToThink o‘zgaruvchisining qiymati NumMinutesToType o‘zgaruvchisi qiymatiga qo‘shiladi. Keyin esa hosil qilingan yig‘indi 60 ga ko‘paytiriladi. Bundan keyin PeopleInTheOffice o‘zgaruvchi qiymati PeopleOnVocation qiymatiga qo‘shiladi. Keyin esa sekundlar soni kishilar soniga ko‘paytiriladi.

Munosabat operatorlari

Bunday operatorlar ikkita qiymatni teng yoki teng emasligini aniqlash uchun ishlatiladi. Taqqoslash ifodasi doimo true (rost) yoki false (yolg‘on) qiymat qaytaradi. Munosabat operatorlarining qo‘llanilishiga oid misol 4.1. jadvalda keltirilgan.

4.1-jadval. C++ tilida ishlatiladigan amallar (operatorlar) ularning tasniflari

Operator	Tavsifi
::	Ko‘rinish sohasiga ruxsat berish
[ ]	Massiv indeksi
( )	Funksiyani chaqirish
.	Struktura yoki sinf elementini tanlash
->	
++	Postfiks inkrement
- -	Postfiks dekrement
++	Prefiks inkrement
- -	Prefiks dekrement
sizeof	O‘lchamni olish
()	Turga akslantirish
~	Razryadli mantiqiy INKOR
!	Mantiqiy inkor
-	Unar minus
+	Unar plyus
&	Adresni olish
*	Vositali murojaat
new	Dinamik ob’yektni yaratish
delete	Dinamik ob’yektni yo‘q qilish
casting	Turga keltirish
*	Ko‘paytirish
/	Bo‘lish
%	Bo‘lish qoldig‘i
+	Qo‘shish
-	Ayirish
>>	Razryad bo‘yicha o‘ngga surish
<	Kichik
<=	Kichik yoki teng
>	Katta
>=	Katta yoki teng
==	Teng

!=	Teng emas
&	Razryadli VA
^	Razryadli istisno qiluvchi YOKI
|	Razryadli YOKI
&&	Mantiqiy VA
||	Mantiqiy YOKI
?:	Shart amali
=	Qiymat berish
*=	Ko‘paytirish qiymat berish bilan
/=	Bo‘lish qiymat berish bilan
%=	Modulli bo‘lish qiymat berish bilan
+=	Qo‘shish qiymat berish bilan
- =	Ayirish qiymat berish bilan
<<=	CHapga surish qiymat berish bilan
>>=	O‘ngga surish qiymat berish bilan
&=	Razryadli VA qiymat berish bilan
^=	Razryadli istisno qiluvchi YOKI qiymat berish bilan
| =	Razryadli YOKI qiymat berish bilan
throw	Istisno holatni yuzaga keltirish
,	Vergul

C++ tili dastur tuzuvchisiga amallarning bajarilish tartibini o‘zgartirish imkoniyatini beradi. Xuddi matematikadagidek, amallarni qavslar yordamida guruhlarga jamlash mumkin. C++ tilida qavs ishlatishga cheklov yo‘q.

Listing 4.3. Qavs yordamida amallarni bajarish tartibini o‘zgartirish

```
#include iostream
int main()
{
int x=0, y=0;
int a=3, b=34, c=82;
x=a*b+c;
y=(a*(b+c));
cout<<<x<<'\n'<<<y<<'\n';
}
```

Dasturda amallar ustunligiga ko‘ra *x* qiymatini hisoblashda oldin a o‘zgaruvchi *b* o‘zgaruvchiga ko‘paytiriladi va unga *c* o‘zgaruvchi qiymatiga qo‘shiladi. Navbatdagi ko‘rsatmani bajarishda esa birinchi navbatda ichki qavs ichidagi ifoda — *(b+c)* qiymati hisoblanadi, keyin bu qiymat a ko‘paytirilib, u o‘zgaruvchisiga o‘zlashtiriladi. Dastur bajarilishi natijasida ekranga **x=184 y=348** satrlari chop etiladi.

5-mavzu. C++ kiritish va chiqarish operatorlari va mantiqiy(boolean) bilan ishlash.

C++ dasturlash tili bilan tanishib borishda davom etamiz. Tushunish oson bo‘lishi uchun console muhitida misollar ko‘ramiz. Console muhitida kiritish operatori cin>> orqali amalga oshiriladi. cin>> dan so‘ng istalgan ma’lumot turi e’lon qilingan o‘zgaruvchini joylashtirsak shu o‘zgaruvchi qiymatini qo‘lda kiritishimiz mumkin bo‘ladi.

Listing 5.1. Kiritish operatoriga misol.

```
#include <iostream.h>
using namespace std;
int main() {
int a=0;
cout<<"a sonini kiriting: "; cin>>a;
cout<<" Siz "<<a<<" sonini kiritdingiz "<
return 0;
}
```

Ushbu dasturni ishga tushirganingizda console kursori sizning biror qiymat kiritishingizni kutib turadi va berilgan o‘zgaruvchiga biror qiymat kiritganingizdan so‘ng, keyingi qator amallarini bajarishga o‘tadi. Yuqoridagi misolda 5 sonini kiritgan bo‘lsangiz, “Siz 5 sonini kiritdingiz” natijasini olishingiz mumkin.

Chiqarish operatori. C++ dasturlash tilida cout kalit so‘zi << bilan birgalikda chiqarish operatori yoziladi.

Operator — bu dasturlash tilida aniq bir maqsadga yo‘naltirilgan. Ishni ma’lum qismini ya’ni C++ misol uchun chiqarish operatori, takrorlash operatori, shart operatorlari va bir qanchasini misol keltirsa bo‘ladi. Operatorlardan so‘ng ; bilan yakunlanadi. Ayrim operatorlar asosan {} figurali qavus bilan bilan ishning ma’lum bir vazifani bajaradigan qismi yakunlanadi.

endl — End Line (qator oxiri) bunda kursor qator oxirida bo‘lib shundagina keyin keladigan amal keyingi qatorga tushadi. Shuni unutmaslik kerakki, chiqarish operatoridan keyin ; ni yozish majburiy hisoblanadi.

Listing 5.2. Endl operatorining ishlatilishiga misol.

```
#include <iostream>
using namespace std;
```

```
int main()
{
  cout << "Salom!" << endl;
  cout << endl; // bunday qo‘llanilishi ham mumkin
  cout << "Mening ismim Alisher << std;
  return 0;
}
```

Kundalik hayotimizda biz tez-tez "Ha" yoki "yo‘q" savollariga aniq javob berish mumkin iboralarga duch kelamiz. Masalan. Olma mevami? – Ha! Qo‘ziqorin sizga yoqadimi? –Yo‘q!

"Olma — bu meva"degan gapni ko‘rib chiqamiz. Bu haqiqatmi? Ha! Olma haqiqatan ham mevadir. Yoki "Men Qo‘ziqorinni yaxshi ko‘raman" deb ayting. Agar siz qo‘ziqorinni xushlamasangiz, bu yolg‘on bo‘ladi.

Faqat ikkita mumkin bo‘lgan natijaga ega bo‘lgan shunga o‘xshash holatlar — ha/rost yoki yo‘q/yolg‘on, shu qadar keng tarqalganki, ko‘plab dasturlash tillari ular bilan ishlash uchun maxsus turni — mantiqiy ma'lumotlar turini qo‘shgan. C++ tilida bu bool ma'lumotlarining mantiqiy turi (ingl. «boolean»).

Mantiqiy o‘zgaruvchilar-bu faqat ikkita mumkin bo‘lgan qiymatdan iborat bo‘lgan o‘zgaruvchilar: true (1) va false (0).

Bool kalit so‘zi mantiqiy o‘zgaruvchini e'lon qilish uchun ishlatiladi:

bool b;

Mantiqiy o‘zgaruvchini ishga tushirish yoki o‘zlashtirish operatsiyasini bajarish true yoki false kalit so‘zlari yordamida amalga oshirilishi mumkin:

```
bool b1 = true; // nusxalovchi initsializatsiya
bool b2(false); // to‘g‘ridan – to‘g‘ri initsializatsiya
bool b3 { true }; // uniform- initsializatsiya (C++11)
b3 = false; // o‘zlashtirish amali
```

Minus (-) unar operatorining ishiga o‘xshash, uning yordamida biz raqamni manfiy qilishimiz mumkin, (!) yordamida biz true-ni false-ga va aksincha (false-ni true-ga) o‘zgartirishimiz mumkin:

```
bool b1 = !true; // b1 qiymati- false
bool b2(!false); // b2 qiymati - true
```

Aslida, mantiqiy qiymatlar true yoki false sifatida saqlanmaydi. Ular butun sonlar sifatida qayta ishlanadi: true o‘rniga bi1, false o‘rniga nol saqlanadi.

Shart operatortini ishlatilishida mantiqiy tipdan foydalanish

Ko‘pincha mantiqiy o‘zgaruvchilar if shart konstruksiyasida ishlatiladi:

```
if (shart) operator1;
```

yoki

```
if (shart) operator1;
else operator2;
```

Listing 5.3. shart operatorida mantiqiy tipning qo‘llanilishi

```
#include <iostream>
using namespace std;
 int main()
{
   std::cout << "butun son kiriting: ";
   int x;
   cin >> x;
   if (x == 0)
      cout << "Kiritgan soningiz nolga teng!" <<endl;
   else
      std::cout << "Kiritgan soningiz noldan farq qiladi" <<endl;
   return 0;
}
```

Listing 5.4. Munosabat operatorida mantiqiy tipning qo‘llanilishi

```
#include <iostream>
using namespace std;

int main() {
  int x = 10;
  int y = 9;
  cout << (x > y);
  return 0;
```

}

Natija:1

5.1-masala. Shaxmat doskasining ikkita turli (x1, y1) , (x2, y2) koordinatalari berilgan (1-8 oraliqda yotuvchi butun sonlar). “Shoh bir yurishda bir maydondan ikkinchisiga o‘ta oladi” jumlani rostlikka tekshiring.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<math.h>
using namespace std;
int main(){
int x1, y1, x2, y2;
bool d;
cout << "x1="; cin >> x1;
cout << "y1="; cin >> y1;
cout << "x2="; cin >> x2;
cout << "y2="; cin >> y2;
d = (abs(x1 — x2) == **1** && y1 == y2) or (abs(y1 — y2) == **1** && x1 == x2)
or (abs(y1 — y2) == **1** && abs(x1 — x2) == **1**);
cout << "Jumlaning rostlik qiymati " << d;
}
	Koordinatalarni kiriting:
	x1=4
	y1=8
	x2=6
	y2=7
	jumlaning rostlik qiymati 0 ga teng

5.2-masala. Temperatura T_f Farangeytda berilgan. Temperatura qiymatini T_c Selsiyga o‘tkazuvchi dastur tuzilsin. $T_c = (T_f - 32) * \frac{5}{9}$

Dastur kodi:	Dastur natijasi:
#include<iostream> **using namespace** std; **int main**() { **double** Tf, Ts; cout<<"temperaturaning farangeytdagi qiymati = "; cin>>Tf;	Tf ni kiriting: 45 Ts=7.22222

```
Ts=((Tf-32)*5)/9;
cout<<"temperaturaning selsiydagi qiymati = ";
cout<<Ts;
}
```

5.3-masala. Qaysidir yil berilgan. Berilgan yilning qaysi yuz yillikka kirishini aniqlovchi programma tuzilsin. (Masalan:20-yuz yillikning boshi 1901-yil)

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main()
{
int n;
cin>>n;
cout<<(n-**1**)/**100**+**1**;
}
	n ni kiriting:
	2005
	21

5.4-Masala. Hafta kunlari quyidagicha tartibda berilgan. 0-yakshanba, 1-dushanba, 2-seshanba, 3-chorshanba, 4-payshanba, 5-juma, 6-shanba. 1-365 oraliqda yotuvchi K soni berilgan, Agar 1-yanvar dushanba bo‘lsa, kiritilgan K-kun haftaning qaysi kuniga to‘g‘ri kelishini aniqlovchi dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main()
{
int k;
cin>>k;
cout<<k%**7**;
}
	k ni kiriting:
	45
	3 kuniga to‘g‘ri keladi.

5.5-masala. Asilbek shunday ikkita a,b sonlarni oldiki, (10*27+22-3)*7-2022=a+b+1 tenglik bajariladi. Ammo noxosdan,

Asilbek bu sonlardan birini yo‘qotib qo‘ydi. Lekin aynan qaysisini yo‘qotganini bilmaydi.

Siz Asilbek yo‘qotgan sonni toping. Yagona qatorda bitta butun son Asilbek yo‘qotgan sonning qiymati kiritiladi. U son modul jihatdan 10^9 dan katta emas. Sizning vazifangiz ekranda Asilbek yo‘qotib qo‘ygan sonni chiqaring.

Dastur kodi:	Dastur natijasi:
#include<iostream> **using namespace** std; **int main**(){ **int** a; cin>>a; cout<<-**1***a;}	a ni kiriting: 13 -13

5.6-masala. N ta talaba K ta niqob sotib oldilar va niqoblarni teng bo‘lib olishga kelishdilar. Har bir talaba nechtadan niqob olishini aniqlaydigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main()
{
 int n,k;
 cin>>n>>k;
 cout<<k/n;
 return 0;
}
	n ni kiriting:
	4
	k ni kiritng:
	15
	3

5.7-Masala. Otabek jamoat transporti uchun chipta sotib olish uchun shaxobchaga bordi. Transport agentligi tomonidan chegirmali chipta e’lon qilinganligi haqida xabar topdi.Omadli chipta bo‘lishi uchun chiptaning raqami 6 xonali bo‘lishi va birinchi 3 ta raqamining yig‘indisi oxirgi 3 ta raqamining yig‘indisiga teng bo‘lishi kerak ekan. Sizning vazifangiz Omadli chipta dasturini tuzish.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main()
{
int n,a,b,c,d,f,g;
cin>>n;
a=n/100000;
b=((n/10000)%10);
c=((n/1000)%10);
d=((n/100)%10);
f=((n/10)%10);
g=n%10;
if((a+b+c)==(d+f+g))
cout<<"OAMDLI CHIPTA";
else cout<<"oddiy chipta";
}
	n ni kiriting:
	145235
	Omadli chipta

5.8-masala. Koshini maktabda matematika darsida o‘qituvchisi doskaga chiqardi va unga 2 ta a, b sonlarni aytdi. Koshi masala shartiga ko‘ra ushbu 2 ta sonni o‘rta arifmetigini va shu sonlarni o‘rta geometrigini hisoblab natijalarning qaysi biri kattaligini hisoblashi kerak. U bu masalani yechishga biroz qiynalyapti. Siz bu masalani yechishda unga yordam bering.

Dastur kodi:	**Dastur natijasi:**

#include<iostream>
using namespace std;
int main(){
long long a,b;
cin>>a>>b;
if(a==b) cout<<"=";
else cout<<">";
}
	a va b sonlarini kiriting:
	a=45 b=43
	>

5.9-masala. Dekard koordinatalar sistemasidagi A(A_x, A_y) va B(B_x, B_y) nuqtalarning koordinatalari berilgan. Shu nuqtalardan hosil bo‘lgan hosil bo‘lgan kesmaning A nuqtasini B nuqta atrofida soat

strelkasi bo‘ylab 180° burgandan keying A nuqtaning koordinatalarini aniqlaydigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int t,x1,x2,y1,y2;
cin>>t;
for(int i=0; i<t; i++){
cin>>x1>>y1>>x2>>y2;
cout<<2*x2-x1<<" "<<2*y2-y1<<"**\n**";
}
}
	Testlar sonini kiriting:
	1
	x1=4
	y1=3
	x2=7
	y2=8
	10 13

5.10-masala. Javohir matematikani yoshligidan yaxshi biladi, ammo ustozi Husayn bergan quyidagi savolga dastur tuzishda qiynalmoqda. Husayn A va B sonlarini o‘yladi ammo Javoxirga Bu sonlarning EKUBI va EKUKI ni beradi, Javoxir shu o‘ylangan sonlar ko‘paytmasini topib beruvchi dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int a,b,y;
cin>>a>>b;
y=a*b;
cout<<y; **return** 0; }
	ekub va ekuk larini kiriting:
	a=11
	b=4
	44

5.11-masala. a so‘mlik pulni 1 so‘mlik hamda 2 so‘mlik pullar yordamida eng kamida necha xil usulda to‘lash mumkin?

Dastur kodi:	Dastur natijasi:
#include<iostream> **using namespace** std; int main(){ int a; cin>>a; cout<<(a/2)+1;	a ni kiriting: 45 23 xil usulda

return 0;}	

5.12-masala. Raqamlari yig‘indisi N ga teng bo‘lgan eng kichik natural sonning 1-raqami chop etilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
long long n;
cin>>n;
cout<<n%9;
}
	n ni kiriting:
	134
	8

5.13-masala. Bir sutkadagi ikki vaqt ko‘rsatkichlari berilgan. Ikkinchi ko‘rsatilgan vaqt birinchi ko‘rsatilgan vaqtdan oldin kelmaganligi aniq. Ikki vaqt ko‘rsatkichlari oralig‘ida necha sekund borligini aniqlang. Elementlarni kiritayotganda 2 qatorda soat(h), minut(m), va sekund(s) lar berilgan bo‘ladi.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<cmath>
using namespace std;
int main(){
int h,m,s,a,b,h1,m1,s1;
 	
cin>>h>>m>>s>>h1>>m1>>s1;
a=h*3600+m*60+s;
b=h1*3600+m1*60+s1;
 	
cout<<abs(a-b);
}
	h , m , s,h1,m1,s1 larni kiriting:
	h=2
	m=6
	s=7
	h1=9
	m1=4
	s1=5
	25078

6-mavzu. C++ da shart operatorlari.

Odatda, dastur ketma-ket bajariladi. *If* ifodasi shartni tekshirishga imkon beradi (masalan, ikkita o‘zgaruvchining tengligi) va taqqoslash natijasiga bog‘liq bo‘lgan boshqa yo‘naltirish orqali dasturning bajarilishini o‘zgartiradi.

If operatorining eng oddiy ko‘rinishi quyidagicha:

```
If(shart)
ifoda;
```

Qavs ichidagi shart har qanday tekshirilishi kerak bo‘lgan ifoda bo‘lishi mumkin, ammo u odatda munosabatlar operatorlarini o‘z ichiga oladi. Agar bu ifoda noto‘g‘ri bo‘lsa, keyingi operator o‘tkazib yuboriladi. Agar u rost qiymatini qaytarsa, u holda operator bajariladi. Quyidagi misolda shart operatoriga misol ko‘rib chiqilgan:

```
if (bigNumber > smallNumber)
bigNumber = smallNumber;
```

Bu yerda bigNumber va smallNumber o‘zgaruvchilarining qiymatlari taqqoslanadi. Agar bigNumber o‘zgaruvchisining qiymati kattaroq bo‘lsa, unda ushbu dastur bo‘lagining ikkinchi qatorida uning qiymati smallNumber o‘zgaruvchisining qiymatiga teng bo‘ladi.

Qavsga olingan ifodalar bloki bitta ifodaga teng bo‘lganligi sababli, bu xususiyat if operatori bilan satr ortida juda keng bo‘lishi mumkin bo‘lgan butun bloklardan foydalanishga imkon beradi:

```
if(shart)
{
ifoda1;
ifoda2;
ifoda3
}
```

Ko‘pincha dasturlarda ma’lum bir shart bajarilganda (ya’ni, ushbu shart rost qiymatini qaytarganda) dastur bitta buyruqlar blokini bajarishi va bajarilmasa (ya’ni, ushbu shart noto‘g‘ri qiymatni qaytarganda) boshqa blokni bajarishi talab qilinadi.

Bir qator shartlarni tekshirish uchun bir nechta if ifodalarini ketma-ket ishlatishning yuqorida ko‘rsatilgan usuli juda samaralidir. Bunday hollarda dasturning o‘qilishini yaxshilash uchun siz *else* kalit so‘zidan foydalanishingiz mumkin.

```
if (shart)
ifoda;
```

```
else
ifoda;
```

Agar shart rost bo‘lsa, unda ifoda bajariladi va undan keyin keyingi ifoda bajariladi. Agar shart bajarilmasa, ya’ni shart false bo‘lsa, unda ifoda e’tiborga olinmaydi va dastur keyingi ifodani bajarishga o‘tadi.

Shuni yodda saqlash tutish lozimki, ifoda o‘rniga figurali qavsga olingan butun blok ishlatilishi mumkin.

Misol:

```
1: if(SomeValue < 10);
2: cout << "SomeValue is less than 10";
3: else
4: cout << "SomeValue is not less than 10!";
5: cout << "Done," << endl;
```

Yuqorida keltirilgan misoldagi 5 satr biz kiritgan shartga umuman bog‘liq emas. Ammo shartning rost yoki yolg‘on yoki rostligiga qarab 2 yoki 4 satrda keltirilgan kodlar ishlashi mumkin.

ICHMA-ICH SHART OPERATORLARI

If-else blokidagi ifoda bloklaridagi har qanday operatorlardan foydalanishda yana qo‘shimcha if va else operatorlaridan foydalanishda hech qanday cheklovlar yo‘q. Misol sifatida keyingi satrlarda bir nechta if operatorlari ichma-ich joylashtirishi keltirilgan:

```
if (shart1)
{
if (shart2)
ifoda1;
else
{
if (shart3)
ifoda2;
else
ifoda3;
}
}
else
ifoda4;
```

Ushbu dasturni quyidagicha tushunish mumkin: agar shart1 to‘g‘ri va shart2 to‘g‘ri bo‘lsa, ifoda1 ni bajaring. Agar shart1 rost bo‘lsa va shart2 rost bo‘lmasa, shart3 ni tekshiriladi va agar rost bo‘lsa, ifoda2 ni

bajariladi. Agar shart1 to‘g‘ri bo‘lsa va shart2 va shart3 bo‘lmasa, unda ifoda3 ni bajariladi. Nihoyat, agar shart1 noto‘g‘ri bo‘lsa, ifoda 4 bajariladi.

Mantiqiy shartlar:

Kichik: a<b

Kichik yoki teng: a<=b

Katta: a>b

Dan katta yoki teng: a>=b

a==b ga teng

Teng emas: a!=b

Turli qarorlar uchun turli harakatlarni bajarish uchun ushbu shartlardan foydalanishingiz mumkin.

C++ da quyidagi shartli bayonotlar mavjud:

- if — belgilangan shart rost bo‘lsa, kod blokda berilgan ifodani hisoblaydi;
- else — agar shart noto‘g‘ri bo‘lsa, bajarilishi kerak bo‘lgan kod blokini belgilash uchun foydalaniladi;
- else if — agar birinchi shart noto‘g‘ri bo‘lsa, sinov uchun yangi shartni belgilash uchun foydalaniladi;
- switch — bajariladigan ko‘plab muqobil kod bloklarini belgilash uchun foydalaniladi.

Shart operatorida doimiy ravishda bloklardan foydalanish xatoliklarni oldini oladi. Ba’zi dasturchilar oldin ochuvchi va yopuvchi qavslarni { , } yozish, undan keyin blok ichidagi operatorlarni yozish lozimligini ta’kidlashadi.

6.1-Listing. Berilgan a sonini juft yoki toqligini aniqlovchi dastur

```
#include<iostream>
using namespace std;
int main(){
int a;
cin>>a;
if(a%2==0)
{cout<<”juft”;
}
else {
cout<<”toq”;
```

```
}
return 0;
}
```

Operatorlar operandlar soniga ko‘ra 3 turga bo‘linadi:

Unar	Binar	Ternar

Ternar operatori – A ? X : Y

Agar A ifoda rost qiymat qaytarsa X ifoda bajariladi, aks holda ya’ni yolg‘on qiymat qaytarsa Y ifoda bajariladi.

Agar tekshirilayotgan shart nisbatan sodda bo‘lsa, shart amalini <<?:>> ko‘rinishini ishlatish mumkin. Bu operator quyidagi ko‘rinishga ega:

<shart ifoda> ? <ifoda1> : <ifoda2>;

if shart operatoriga o‘xshash holda bu shart amali quyidagicha ishlaydi:agar <shart ifoda > rost (true) bo‘lsa (ifoda1) bajariladi, aks holda <ifoda2>. Odatda ifodalar qiymatlari birorta o‘zgaruvchiga o‘zlashtiriladi

Masalan:

Z=(X<Y) ? X+15 : Y-25

X=15 Y=12

Z=-13

Izoh: Yuqoridagi masalada agar berilgan X soni Y sonidan kichik bo‘lsa X+15 ifoda, aks holda Y-25 ifoda bajarilsin deyilgan. Biz bergan namunada X qiymat Y qiymatdan katta bo‘lganligi sababli , ya’ni aks holda qismi ishlagani sababli Y-25 ifoda bajarildi va natija chiqarildi.

Mantiqiy operatorlar.

Mantiqiy toifa *bool* ikki turdagi qiymat qabul qilishi mumkin: true (rost,1) va false (yolg‘on, 0). Mantiqiy ma’lumotlari e’lon qilish uchun *bool* xizmatchi so‘zidan foydalaniladi.

Mantiqiy ifoda 1 (to‘g‘ri) yoki 0 (noto‘g‘ri) bo‘lgan mantiqiy qiymatni qaytaradi. Bu mantiqni shakllantirish va javoblarni topish uchun foydalidir.

Ifodaning (yoki o‘zgaruvchining) to‘g‘ri yoki noto‘g‘ri ekanligini aniqlash uchun () dan katta operator kabi taqqoslash operatoridan foydalanishingiz mumkin;

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
  float s, a, p, oy = **0**;
  cout << "s="; cin >> s;
  cout << "p="; cin >> p;
  a = s;
  while(a < **2** * s)
  {
  a += p * a / **100**;
  oy++;
  cout << oy << ")" << a << endl;
  }
  cout << oy << " oydan keyin " << a;
}
	1

Bool a, b;

Mantiqiy toifadagi o‘zgaruvchilarga qiymat berish quyidagicha amalga oshiriladi:

a = true; *// rost*

b = false; *//yolg‘on*

Mantiqiy amallar:

- !(inkor qilish) – mantiqiy operatori mantiqiy ifodalar yoki o‘zgaruvchilar oldidan qo‘yiladi. Mantiqiy ifoda yoki o‘zgaruvchining qiymatini teskarisiga o‘zgartiradi;

- && (mantiqiy ko‘paytirish) – mantiqiy operatori ikkita mantiqiy o‘zgaruvchini birlashtiradi. Agar ikkala o‘zgaruvchi ham rost qiymatga ega bo‘lsa natija rost, aks holda yolg‘on natija beradi;

- || (mantiqiy qo‘shish) – mantiqiy qo‘shish operatori ikkita mantiqiy o‘zgaruvchini birlashtiradi. Agar o‘zgaruvchilardan kamida bittasi rost qiymatga ega bo‘lsa natija rost, aks holda yolg‘on natija beradi.

Shart operatori va mantiqiy amallarga doir masalalar.

6.1-masala. N natural soni berilgan. Sizning vazifangiz N ta tomonga ega bo‘lgan qavariq ko‘pburchakning diagonallar sonini topishdan iborat.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
long long n;
cin>>n;
if(n<**4**) cout<<**0**;
else
cout<<(n*(n-**3**))/**2**;
return 0;}
	N ni kiriting:
	8
	Diagonallar soni= 20

6.2-masala. Ikki natural sonning eng katta umumiy bo‘luvchisini topadigan dastur tuzing. Bu dasturni rekursiya orqali amalga oshiring.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int kir(**int** a,**int** b) {
if(a!=b) {
if (a > b) a %= b;
else b %= a;
if (a == **0**) a = b;
if (b == **0**) b = a;
kir(a,b);
} **else return** a;
}int main() {
int x,y;
cin>>x>>y;
cout<<kir(x,y);
}
	x va y ni kiriting:
	x=25 y=45
	EKUB=5

6.3-masala. N natural soni berilgan. Bu sonning tub yoki tub emasligini aniqlaydigan dastur tuzing. Agar son tub bo‘lsa ekranga “tub” , aks holda “tub emas” yozuvi chiqarilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
  int n;
  bool b=true;
  cout<<"n ni kiriting =";
  cin>>n;
  for(**int** i=**2**; i<n; i++){
  if(n%i==**0**){
  b=false;
  break;
  }
  }
  if(b){
  cout<<n<<" tub son";
  }
  else{
  cout<<n<<" tub son emas";
  }
  return 0;
}
	n ni kiritng:
	16
	16 tub son emas

6.4-masala. Ikki natural sonning eng kichik umumiy karralisini topadigan dastur tuzing. Bu dasturni rekursiya va rekursiyasiz holatda amalga oshiring.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int kir(**int** a,**int** b) {
  if(a!=b) {
  if (a > b) a %= b;
  else b %= a;
  if (a == **0**) a = b;
  if (b == **0**) b = a;
  kir(a,b);
  } **else return** a;
	a va b ni kiriting:
	a=81 b=12
	EKUK=324

```
}int main() {
   int x,y;
   cin>>x>>y;
   cout<<x*y/(kir(x,y));
}
```

6.5-masala. Uchta son berilgan. Bunday uzunlikdagi segmentlardan uchburchak hosil qilish mumkinligini aniqlaydigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int a,b,c;
cout<<”uchburchak tomonlarini kiriting : “<<endl;
cout<<”a=”;
cin>>a;
cout<<”b=”;
cin>>b;
cout<<”c=”;
cin>>c;
if((a+b>c) && (a+c>b) && (b+c>a))
cout<<”uchburchak hosil qilib bo‘ladi”;**else** cout<<”uchburchak hosil qilib bo‘lmaydi”;
}
	uchburchak tomonlarini kiriting:
	a=3
	b=4
	c=5
	uchburchak hosil qilib bo‘ladi.

6.6-masala. Koordinata o‘qlariga parallel ravishda to‘g‘ri to‘rtburchakning uchta uchi berilgan, to‘rtinchi uchi koordinatasini aniqlaydigan dastur tuzilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int x1, y1, x2, y2, x3, y3, x4, y4;
cout << "x1="; cin >> x1;
	koordinata qiymatlarini kiriting:
	x1=4
	y1=0
	x2=4

cout << "y1="; cin >> y1;
cout << "x2="; cin >> x2;
cout << "y2="; cin >> y2;
cout << "x3="; cin >> x3;
cout << "y3="; cin >> y3;
if(x1 == x2) x4 = x3;
else if(x1 == x3) x4 = x2;
else if(x2 == x3) x4 = x1;
if(y1 == y2) y4 = y3;
else if(y1 == y3) y4 = y2;
else if(y2 == y3) y4 = y1;
cout << "4-nuqta " << "(" << x4 << "," << y4 << ")";
}
	y2=3
	x3=0
	y3=3
	4-nuqta (0,0)

7-mavzu. C++ da takrorlash operatorlari.

Agar siklning takrorlanishlar soni oldindan ma’lum bo‘lmasa, ya’ni hisoblash jarayonida qaysidir o‘zgaruvchining qiymatiga bog‘liq bo‘lsa u holda **do-while** yoki **while** operatoridan foydalaniladi.

Belgilangan shartga erishilganda, sikllar kod blokini bajarishi mumkin.

Sikllar qulay, chunki ular vaqtni tejaydi, xatolarni kamaytiradi va kodni o‘qishni osonlashtiradi.

C++ while sikli

while Belgilangan shart mavjud bo‘lganda, sikl kod bloki bo‘ylab aylanadi: Sintaksisi:

```
while (shart) {
  // kod bloki bajariladi
}
```

Quyidagi misolda, o‘zgaruvchi 5 dan kichik bo‘lsa, sikldagi kod qayta-qayta ishlaydi:

```
int i = 0;
while (i < 5) {
  cout << i << "\n";
  i++;
}
```

Eslatma: Shartda ishlatiladigan o‘zgaruvchini oshirishni unutmang, aks holda sikl cheksiz davom etadi.

do-while operatori siklda oldin sikl tanasini bir marta bajarib, so‘ngra tekshirish amali bajariladi. Uning sintaksisi umumiy ko‘rinishi:

do{
operator;
}while(shart);

while takrorlash operatori, operator yoki blokni takrorlash sharti yolg‘on, (false yoki 0) bo‘lguncha takror bajaradi. U quyidagi sintaksisga ega.

while (<ifoda >) <operator>;

Agar <ifoda> rost qiymatli o‘zgarmas qiymat bo‘lsa, takrorlash cheksiz bo‘ladi. <ifoda> takrorlash tanasidagi rost bo‘lib, uning qiymatiga tanasidagi hisoblash ta’sir etmaydi.

do-while operatorida siklning tanasi kamida bir marta takrorlanadi. Shu bir marta hisoblash ham yechilayotgan masalani

mohiyatini buzib yuborishi mumkin. Bunday holatlarda **while** sikl operatoridan foydalangan maqsadaga muvofiq hisoblanadi.

while operatori sikl tanasi ixtiyoriy operator yoki operatorlar majmuidan iborat bo‘lishi mumkin.

Agar shart **rost** qiymatga ega bo‘lsa, sikl tanasi bajariladi, aks holda, ya’ni shart yolg‘on qiymatga teng bo‘lsa sikl tugatiladi.

While sikl operatoridan qachon (shart) yolg‘on qiymat qabul qilsa chiqiladi. Ya’ni while operatoridan keyingi operatorga uzatiladi. Agar (shart) false qiymat qabul qilmasa, while sikl operatoridan chiqib ketilmaydi va bu jarayon **sikllanib qolish** deyiladi.

do xizmatchi so‘zidan keyingi operatorlar bajariladi, keyin **while** xizmatchi so‘zidan keying shart tekshiriladi. Agar shart rost bo‘lsa do xizmatchi so‘zidan keyingi operatorlar qayta bajariladi. Shart qayta tekshiriladi, bu jarayon shart yolg‘on natija berguncha takrorlanadi. Agar while xizmatchi so‘zidan keyingi shart yolg‘on bo‘lsa, boshqarilish do-while operatoridan keyingi operatorga uzatiladi.

do-while va **while** sikl operatorlarida sikl tanasi sifatida faqat bitta operator ishlatiladigan bo‘lsa, bu operatorni blok orasiga {} olmasdan ham yoziish mumkin.

Lekin professional dasturchilar har qanday holda sikl tanasini blokka {} olib yozishni tavsiya qilishadi. Bu esa sodir bo‘lishi mumkin bo‘lgan mantiqiy xatoliklarni oldini oladi.

7.1-Masala. n natural soni berilgan (n>1). (1+2+3+……+k)>=n shart bajariladigan eng kichik k sonini aniqlovchi dastur tuzilsin. 1 dan k gacha bo‘lgan yig‘indi ham ekranga chiqarilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
  int n, k = **0**, s = **0**;
  cout << "n="; cin >> n;
  while(n > s)
  {
  k++;
  s += k;
  }
  cout << "k=" << k << " " << "s=" << s;
  }
	N ni kiriting:
	5
	K=3 s=6

7.2-masala. A, B, C musbat butun sonlari berilgan. A x B to‘rtburchak ichida tomoni C bo‘lgan kvadratdan nechtasi sig‘ishini aniqlovchi dastur tuzilsin. Ko‘paytirish va bo‘lish amallarini ishlatmang.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<math.h>
using namespace std;
int main(){
int a, b, c, boyi = **0**, ans = **0**;
cout << "a="; cin >> a;
cout << "b="; cin >> b;
cout << "c="; cin >> c;
while(a >= c)
{
a -= c;
boyi ++;
}
while(b >= c)
{
ans += boyi;
b -= c;
}
cout << ans; }
	a=8 b=7 c=2
	12

7.3-masala. Satr berilgan. Satrdagi ortiqcha probellarni olib tashlovchi dasturi tuzilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<string>
using namespace std;
int main(){
string s,ans="";
getline(cin,s);
for(**int** i=**0**; i<s.size(); i++)
{
	Satrni kiriting:
	Salom dunyo
	Salom dunyo

```
        while(s[i]==' ' && s[i+1]==' ')
        i++;
        ans+=s[i];
                }
        cout<<ans;
}
```

7.4-masala. k natural soni va k to‘plam berilgan. Har bir to‘plam elementlarini kiritish 0 bilan tugaydi. Har bir to‘plam kamida 2 ta elementga ega. Elementlari o‘suvchi bo‘lgan to‘plamlar sonini chiqaruvchi dastur tuzing.

Dastur kodi:

```
#include<iostream>
using namespace std;
int main(){
        int k, s=0;
        cout<<"To‘plamlar sonini kiriting k=";
        cin>>k;
        float a, a1;
        for(int i=0; i<k; i++)
        {
                bool osuvchimi=true;
                cout<<i+1<<"-to‘plam elementlarini
kiriting\n";
                cin>>a;
                do
                {
                        cin>>a1;
                        if(a1<=a && a1!=0)
osuvchimi=false;
                        a=a1;
                }
                while(a1!=0);
                s+=osuvchimi;
        }
        cout<<k<<" ta to‘plam ichida o‘suvchilar
soni"<<s;
}
```

Dastur natijasi:

```
        To‘plamlar sonini
kiriting k=2
        1-to‘plam
elementlarini kiriting
        1
        2
        0
        2-to‘plam
elementlarini kiriting
        3
        5
        7
        0
        2 ta to‘plam ichida
o‘suvchilar soni2
```

7.5-masala. Sizga butun sonlar to‘plami berilgan. To‘plamda 1 ta elementdan tashqari barchasini jufti bor. To‘plamdagi yagona jufti bo‘lmagan yolg‘iz sonni topadigan dastur tuzing. Masalan:[1,2,3,,4,3,2,1] to‘plamdagi yolg‘iz son 4 sonidir.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;int s[101];int main(){
int n,a;
cin>>n;
for(int i=0; i<n; i++){
cin>>a;
s[a]++;
}
a=0;
while(s[a]%2==0) a++;
cout<<a;
}
	N ni kiriting:
	7
	1 2 3 4 3 2 1
	4

For sikl operatori

Quyida **for** sikli sintaksis ko‘rinishi berilgan:

```
for (shart 1; shart 2; shart 3) {
  // for sikl tanasi }
```

Misol ko‘rib chiqamiz. Takrorlash Operatorida berilgan misoldagi kodni for tskilda yozamiz.

```
#include <iostream>
using namespace std;

int main() {
  for (int i = 0; i <= 7; i++) {
    cout << i << "\n";
  }
  return 0;
}
```

Natijada ekranda 1 dan 7 gacha bo‘lgan sonlar ekranga chiqariladi.

Misolni tahlili: 1 — shart: sikl aylanishi jarayonida bog‘liq bo‘lgan o‘zgaruvchi qiymati beriladi. int i = 0;

2 — shart: tskilni aylanish davrini belgilaysiz. bunda asosan boradigan son beriladi. Shuni yodda saqlash lozimki, sikl qadamiga teskari son qo‘yilganda tskil tugamaydi.

3 — shart: sikl qadami bo‘lib har bir aylangan vaqtda sikl bo‘gliq bo‘lgan o‘zgaruvchining oshib boradigan qiymati tushuniladi.

Misol: 0 dan 10 gacha bo‘lgan sonlarni faqat juft sonlarni chiqarish dasturini tuzish.

```
#include <iostream>
using namespace std;

int main() {
  for (int i = 2; i <= 10; i = i + 2) {
    cout << i << "\n";
  }
  return 0;
}
```

Natija sifatida ekranga 2 4 6 8 10 raqamlari chiqadi.

7.6-Masala. Natural n sonining faktoriali deb 1 dan n gacha bo‘lgan sonlar ko‘paytmasiga aytiladi. Sizga biror bir n natural soni berilgan. Berilgan n sonining faktorialini chiqaradigan dasturni tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main() {
int n,sum=1;
cout<<”n ni kiriting=”;
cin>>n;
for(int i=1; i<=n; i++) {
sum*=i;
}
cout<<n<<” faktorial=”<<sum;
}
	n ni kiriting:
	5
	5 faktorial = 120

7.7-Masala. 1,1,2,3,5,8,… qatori Fibonachchi qatori deyiladi. Uchinchi elementdan boshlab har bir element oldingi ikkitasini yig‘indisiga teng bo‘lsa bunday sonlar Fibonachchi sonlar deyiladi. Fibonachchi qatorining n-sonini topadigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int n, f1 = **1**, f2 = **1**, f;
cin >> n;
for(**int** i = **1**; i <= n; i++)
{
if(i == **1** or i == **2**) f = **1**;
else f = f1 + f2;
f1 = f2;
f2 = f;
}
cout << f << " ";
}
	n ni kiriting:
	9
	9-fibonachchi soni=34;

7.8-masala. Berilgan formulalar orqali ε va π sonlarining taxminiy qiymatini topadigan dastur tuzing.

Bunda $e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots$ va $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \cdots$

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main() {
double i, fakt=**1**, e=**0**, p=**1**;
for(i=**2**;i<**8**;i++) {
e+=**1**/fakt;
fakt*=i; }
bool b=**1**;
for(i=**3**;i<**10000**;i+=**2**){
if(b)p-=**1**/i;
else p+=**1**/i;
b=!b;
}
cout<<e<<" "<<p***4**;
}
	e va p sonlarining taxminiy qiymatlari:
	e=1.71806
	p=3.14139

7.9-masala. Agar biror-bir natural n sonning kvadrati birinchi kelgan n ta toq sonlar yig‘indisiga teng bo‘lsa, berilgan n sonini kvadratini yig‘indi orqali hisoblaydigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include <iostream>
using namespace std;
int main() {
int n;
cin >> n;
int sum = **0**;
for (**int** i = **1**; i <= **2***n-**1**; i += **2**) {
sum += i;
cout << i <<” + “<<” “;
}
cout<<” = “<<sum;
return 0;
}
	n ni kiriting:
	6
	1+3+5+7+9+11=36

7.10-masala. To‘g‘ri to‘rtburchak shaklidagi maydonning yuzasi berilgan. Bu maydonni hosil qilish uchun maydon tomonlari qiymatlarni necha xil usulda tuzish mumkinligini aniqlaydigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main() {
long c=**0**,n;
cin>>n;
for(**int** i=**1**;i*i<=n;i++){
if (n%i==**0**) c++; }cout<<c;
}
	maydon yuzasini kiriting:
	4
	2 xil usulda tuzish mumkin.

7.11-Masala. Ikki natural sonning kublari yig‘indisi sifatida ifodalash mumkin bo‘lgan, eng kichik natural sonni topadigan dastur tuzing.

Dastur kodi:	Dastur natijasi:
#include<iostream> #include<cmath> **using namespace** std; **int main**(){ **int** n; cin>>n;	n ni kiriting: 45 eng kichik natural son:54

```
bool b=false;
while(!b){
for(int i=1; i<=cbrt(n);i++){
int j=round(cbrt(n-pow(i,3)));
if(i*i*i+j*j*j==n){
cout<<"eng kichik natural son="<<n; b=true;
break;
                }
                }
                n++;
}
return 0;
}
```

712-masala. Agar 100 bosh qoramolni 100 so‘mga sotib olish kerak bo‘lsa, 1ta buqaga 10 so‘m, 1 ta sigirga 5 so‘m va 1 ta buzoqqa 50 tiyin to‘lab , nechta buqa, sigir va buzoq sotib olish mumkinligini aniqlaydigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main() {
// x – buqa soni
//y – sigir soni
// z – buzoq soni
// x*10 + y*5 + z*0.5 = 100
// x + y + z = 100
for(int i=1;i<=100;i++) {
for(int j=1;j<=100-i;j++) {
for(int k=1;k<=100-j-i;k++) {
if(i+j+k==100 && (i*10 + j*5 + k*0.5) == 100){
cout<<i<<" ta buqa\n";
cout<<j<<"ta sigir\n";
cout<<k<<" ta buzoq\n";return 0;
} } }} }
	100 so‘mga
	1 ta buqa
	9 ta sigir
	90 ta buzoq olish mumkin

7.13-masala. N(32<=n<=126) butun soni berilgan. Kodi n ga teng bo‘lgan simvol chop etilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<string>
using namespace std;
int main(){
int n;
cout<<”n=”;
cin>>n;
cout<<”bu butun sonning simvoli: “;
cout<<**char**(n);
cout<<” hisoblanadi.”;
}
	n ni kiriting:
	65
	bu butun sonning simvoli A hisoblanadi.

7.14-Masala. A va B butun soni berilgan (A<B). A va B sonlari orasidagi barcha butun sonlarni chiqaruvchi programma tuzilsin. Bunda har bir son o‘zining qiymaticha chiqarilsin. Ya’ni 3 soni 3 marta, 4 soni 4 marta chiqariladi.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<cmath>
using namespace std;
int main(){
int a, b;
cout << "a="; cin >> a;
cout << "b="; cin >> b;
for(**int** i = a ; i <= b ; i++)
{
for(**int** j = **0**; j < i; j++)
{
cout << i << " ";
}
cout << "\n";
}
}
	a va b sonlarini kiriting:
	a=2 b=5
	2 2
	3 3 3
	4 4 4 4
	5 5 5 5 5

7.15-Masala. X haqiqiy soni berilgan. Quyidagi funksiya hisoblansin.

$$f(x)=\begin{cases}2\sin x,\ agar\ \ x>0\\ x-6,\ \ agar\ \ x\le 0.\end{cases}$$

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<cmath>
using namespace std;
int main(){
  int x;
  cout << "x="; cin >> x;
  if(x>**0**) cout<<**2***sin(x);
  else cout<<x-**6**;
}
	x sonini kiriting:
	1.0
	1.68294

7.16-Masala. Bankka boshlang‘ich S so‘m qo‘yildi. Har oyda bor bo‘lgan summa p foizga oshadi (0<p<25). Necha oydan keyin boshlang‘ich qiymat 2 martadan ko‘p bo‘lishini hisoblovchi dastur tuzilsin. Bankda hosil bo‘lgan summa haqiqiy son ekranga chiqarilsin.

Dastur kodi:	Dastur natijasi:

include<iostream>
using namespace std;
int main(){
  float s, a, p, oy = **0**;
  cout << "s="; cin >> s;
  cout << "p="; cin >> p;
  a = s;
  while(a < **2** * s)
  {
  a += p * a / **100**;
  oy++;
  cout << oy << ")" << a << endl;
  }
  cout << oy << " oydan keyin "
<< a;
}
	s ni kiriting:
	10000
	p ni kiriting:
	15
	1)11500
	2)13225
	3)15208.8
	4)17490.1
	5)20113.6
	5 oydan keyin 20113.6

7.17-Masala. Berilgan n natural sonining barcha bo‘luvchilarini, shuningdek ularning soni va yig‘indisini topib ekranga chiqaradigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int n,k=0,sum=0;
cout<<"n ni kiriting =";
cin>>n;
cout<<"n sonining bo‘luvchilari:\n;
for(int i=1; i<=n; i++){
if(n%i==0){
k++;
sum+=i;
cout<<i<<" ";
} }
cout<<endl;
cout<<"bo‘luvchilar soni "<<k<<" ga teng"<<endl;
cout<<"bo‘luvchilar yig‘indisi "<<sum<<" ga teng"<<endl;
}
	n ni kiriting:
	12
	n sonining bo‘luvchilari:
	1 2 3 4 6 12
	Bo‘luvchilari soni 6 ga teng;
	Bo‘luvchilari yig‘indisi 28 ga teng

7.18-masala. [2 ;100] oralig‘idagi barcha egizak sonlarni topadigan dastur tuzing. Bir-biridan 2 ga farq qiladigan ikkita toq tub sonlar egizak sonlar deyiladi. Masalan, 5 va 7 bunga misol bo‘la oladi.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main() {
int c=0,w=0;
for(int i=2; i<=100; i++) {
bool b=1;
if(i%2==0 && i!=2)b=0;
for(int j=3; j<i; j+=2) {
	5 7
	11 13
	17 19
	41 43
	59 61

```
            if(i%j==0) {
               b=0;
               break;
            }
         }
         if(b) {
            c++;
            if(c==2 &&
w+2==i)cout<<w<<" "<<i<<endl;
            if(c==2)c=0;
            w=i;
         }
 }
}
```

7.19-masala. N soni berilgan. Bu sonning mukammal yoki mukammal emasligini aniqlaydigan dastur tuzing.Mukammal son deb o‘zining barcha bo‘luvchilari yig‘indisiga teng bo‘lgan songa aytiladi. Bunga misol qilib (28=1+2+4+7+14)ni olishimiz mumkin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int n,k=**0**,sum=**0**;
cout<<"n ni kiriting =";
cin>>n;
for(**int** i=**1**; i<n; i++){
if(n%i==**0**){
sum+=i;
} }
if(n==sum)
cout<<"mukammal son";
else cout<<"mukammal son emas"; }
	n ni kiriting :
	45
	Mukammal son emas

7.20-Masala. Chapdan o‘ngga va o‘ngdan chapga bir xil o‘qiladigan sonlarga palindrom sonlar deyiladi. 1223221 soni palindrom songa misol bo‘lishi mumkin.

Berilgan son palindrom yoki palindrom emasligini aniqlaydigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main() {
string s;
cin>>s;
bool b=**1**;
if(s.size()%**2**) {
for(**int** i=**0**; i<=s.size()/**2**; i++)
if(s[i]!=s[s.size()-i-**1**]) {
b=**0**;
break;
}
if(b)cout<<"palindrom";
else cout<<"palindrom emas";
} **else** cout<<"palindrom emas";
}
	s ni kiriting:
	777
	Palindrom son

7.21-masala. Kitob n ta sahifadan iborat . Bunday kitobning barcha sahifalarini raqamlash uchun zarur bo‘lgan raqamlar sonini topadigan dastur kodini yozing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int n;
cout<<"n=";
cin>>n;
if(n%**2**==**0**) {
 	
cout<<**2***n<<" ta son kerak";}
else cout<<**2***n+**1**<<" ta son kerak ";
}
	n ni kiriting:
	88
	176 ta son kerak

7.22-masala. Trolleybus chiptalari raqamlari olti xonali raqamlardir. Agar birinchi uchta raqamlar yig‘indisi oxirgi uchta raqam yig‘indisiga teng bo‘lsa,chipta omadli hisoblanadi. Masalan 627294 chipta omadli, chunki 6+2+7=2+9+4=15. Chipta omadli bo‘lishi uchun chipta raqamini zarur bo‘lgan holatlarini topadigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main() {
int i,j,k,l,m,n;
for(i=1; i<10; i++)
for(j=1; j<10; j++)
for(k=1; k<10; k++)
for(l=1; l<10; l++)
for(m=1; m<10; m++)
for(n=1; n<10; n++)
if(i+j+k==l+m+n)
cout<<i<<j<<k<<l<<m<<n<<endl;
}
	Bu holatlar cheksiz davom etadi, buni kompilyator orqali ishlatib ko‘rishingiz mumkin.

7.23-masala. Raqamlarining kublarining yig‘indisiga teng natural sonlar mavjud, masalan, 370 soni bunga misol, chunki $3^3 + 7^3 + 0^3 = 370$.Shu kabi sonlarni topadigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<cmath>
using namespace std;
int main(){
for(int i=1; i<=1000; i++){
int sum=0;
int t=i;
while(t>0){
int d=t%10;
sum+=pow(d,3);
t/=10;
}
if(sum==i){
cout<<i<<endl;
}
} }
	1
	153
	370
	371
	407

7.24-masala. Gomer Simpsonning tushlik tanaffusi T sekundni tashkil qiladi. Gomer N sekundda bitta gamburger, M sekundda bitta cheesburger yeyishi mumkin. Gomer tushlik paytida maksimal umumiy holatda nechta gamburger va nechta cheesburger yeyishi mumkin bo‘lgan holatni topadigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include <iostream>
using namespace std;
int main() {
int T, N, M;
cin >> T >> N >> M;
int max_hamburgers = 0;
for (int i = 0; i <= T / N; i++) {
int berilgan_vaqt = T — i * N;
int j = berilgan_vaqt / M;
int jami_burgerlar = i + j;
if (jami_burgerlar > max_hamburgers) {
max_hamburgers = jami_burgerlar;
}
}
cout << "Maximal darajada yeyishi mumkin bo‘lgani : " << max_hamburgers << endl;
return 0;
}
	t,n,m sonlarini kiriting:
	45
	12
	2
	Maximal darajada yeyishi mumkin bo‘lgani : 22

7.25-masala. n butun soni berilgan (n>0). Bir sikldan foydalangan holda quyidagi yig‘indini hisoblovchi programma tuzilsin.

1!+2!+3!+…..+n!

Dastur kodi:	Dastur natijasi:

#include <iostream>
using namespace std;
int main() {
int n, sum = 0, fact = 1;
cout << "butun sonni kiriting: ";
cin >> n;
for (int i = 1; i <= n; i++) {
	n ni kiriting:
	4
	1 dan n gacha faktoriallar yig‘indisi 33 ga teng

```
        fact *= i;
        sum += fact;
    }
    cout << "1 dan n gacha faktoriallar yig‘indisi
"  << sum <<" ga teng "<< endl;
    return 0;
}
```

7.26-masala. n butun soni berilgan (n>1). Fibonachchi ketma-ketlikning dastlabki n ta hadini chiqaruvchi programma tuzilsin.

$F_1 = 1$, $F_2 = 1$ $F_K = F_{K-2} + F_{K-1}$; K=3,4,…

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<cmath>
using namespace std;
int main(){
int n, f1 = **1**, f2 = **1**, f;
cout << "n="; cin >> n;
for(**int** i = **1**; i <= n; i++)
{
if(i == **1** or i == **2**) f = **1**;
else f = f1 + f2;
cout << f << " ";
f1 = f2;
f2 = f;
}
}
	N ni kiriting:
	12
	1 1 2 3 5 8 13 21 34 55 89 144

7.27-masala. N natural soni va N ta butun sondan iborat to‘plam berilgan. To‘plamdagi ketma-ket keladigan eng kichik elementlarning maksimal sonini aniqlovchi dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int n, a, min1, soni1, min, soni = **1**;
cout << "n="; cin >> n;
cin >> a;
	N ni kiriting:
	7
	1 5 8 9 12 45 11
	Min=1 soni=1

```
  min1 = a; soni1 = 1;
  for(int i = 2; i <= n; i++)
  {
    cin >> a;
    if(min1 == a)
    {
      soni1 ++;
      if(soni < soni1) soni = soni1;
    }
    if(a < min1 )
    {
      min1 = a; soni1 = soni = 1;
    }
    if(a > min1)
    {
      soni1 = 0;
    }
  }
  cout << "min=" << min1 << " " << " soni=" <<
soni;
}
```

7.28-masala. Quyidagi shartlardan birini bajaradigan yil kabisa yili hisoblanadi:

-Yil raqami 400 ga bo‘linsa

-Yil raqami 4ga bo‘linsa va 100 ga bo‘linmasa.

Sizning vazifangiz agar kiritilgan yil kabisa yili bo‘lsa “Kabisa yili” aks holda “Kabisa yili emas” yozuvini chiqaruvchi dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
  int n;
  cin>>n;
  if ((n%4==0 &&
n%100!=0)||(n%400==0))
    cout<<"Kabisa yili";
  else cout<<"Kabisa yili emas";
}
	N ni kiriting:
	2004
	Kabisa yili

7.29-masala. Kadrlar bo‘limida ish haqqini so‘mda oladigan 3 nafar xodim ishlaydi. Ulardan eng yuqori maosh oluvchining maoshi eng kam maosh oluvchidan qancha farq qilishini topuvchi dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int a,b,c;
cin>>a>>b>>c;
if(a<b && b<c) cout<<c-a;
else if(a<c && c<b) cout<<b-a;
else if(b<a && a<c) cout<<c-b;
else if(b<c && c<a) cout<<a-b;
else if(c<a && a<b) cout<<b-c;
else if(c<b && b<a) cout<<a-c;
return 0;}
	a va b va c sonlarini kiriting:
	a=145000 b=47000
	c=150000
	farq=103000

7.30-masala. Shaxmat musobaqasida N ta jamoa qatnashadi. Agar bitta jamoa bitta sovrinni qo‘lga kirita olsa, oltin, kumush va bronza medallar to‘plamini taqsimlashning nechta variant mavjud?

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
long long n, mod = 1e9+7, sum = 1;
cin>>n;
if(n==1) sum = 1;
else if(n==2) sum = 2;
else{ sum*=n%mod;
sum%=mod;
sum*=(n-1)%mod;
sum%=mod;
sum*=(n-2)%mod;
sum%=mod;}
cout<<sum; }
	n ni kiriting:
	12
	1320 xil variant bor.

8-mavzu. C++da ko‘rsatkichlar va adres oluvchi o‘zgaruvchilar.

Ko‘rsatkichlar C++ dasturlash tilining kuchli qurollaridan biridir. Tushunchaning nomi nega ko‘rsatkich, u nimani ko‘rsatadi degan savollar tug‘ilishi tabiiy. Ko‘rsatkich ham o‘zgaruvchi bo‘lib, u ham e’lon qilinadi va uning oddiy o‘zgaruvchilar e’lonidan farqi shundaki, ularni e’lon qilishga oldidan (*) yulduzcha qo‘yiladi. Ko‘rsatkich o‘zgaruvchi boshqa o‘zgaruvchining EHM xotirasidagi adresiga teng qiymatni oladi, ya’ni o‘sha o‘zgaruvchining adresini ko‘rsatadi. C++ dasturlash tilida tuzilgan dastur tarkibidagi o‘zgaruvchilar dastur kompilyatsiya qilingandan keyin kompyuter xotirasiga borib joylashadi. Kompilyator dastur matnini kompyuter xotirasiga joylashtirgandan so‘ng uning tarkibidagi o‘zgaruvchilar adresini ob’eykt kod sifatida qabul qilib oladi.

Masalan, int *x; yozuvi butun turli x nomli ko‘rsatkich o‘zgaruvchi, float *a; yozuvi esa a nomli haqiqiy turli ko‘rsatkich o‘zgaruvchi e’lon qilinganligini bildiradi. Yoki boshqacha qilib aytganda o‘zgaruvchi e’lon qilinishi chog‘ida uning oldiga (*) belgisi qo‘yilsa, uni ko‘rsatkichga aylantiradi. Bundan buyon «ko‘rsatkich o‘zgaruvchi» o‘rniga «ko‘rsatkich» so‘zidan foydalanamiz.

Shuni yodda saqlash kerakki, ko‘rsatkich qaysi turga mansub bo‘lsa, faqat shu turga taalluqli o‘zgaruvchilarning adreslarini aniqlay oladi.

Ko‘rsatkichlar bilan ikkita, ya’ni «*» va «&» operatorlari qo‘llaniladi. Bunda «&» operatori o‘zgaruvchining EHM xotirasidagi adresini aniqlaydi.

«*» operatori esa «&» aniqlangan o‘zgaruvchining o‘sha adresga joylashgan qiymatini aniqlaydi. Bu fikrlarni quyidagi dastur yordamida tushunib olish mumkin.

```
int main()
{
   int n,m, *x;
   n=423;
   x=&n;
   cout<<x<<endl;
   m=*x;
   cout<<m<<endl;
   return 0;
}
```

C++ dasturlash tilida tuzilgan dastur tarkibidagi o‘zgaruvchilar, o‘zgarmaslar va funksiyalar adreslarini kompyuter xotirasiga alohida saqlash va ular ustida amallar bajarish mumkin.

Ta’rif: Qiymatlari adres bo‘lgan o’zgaruvchilar ko‘rsatkich o’zgaruvchilar deb ataladi.

Ko‘rsatkichlar uch xil turda bo‘ladi:

- Funksiyaga ko‘rsatkich;
- Ob’yekt o‘zgaruvchiga ko‘rsatkich;
- Void ko‘rsatkich.

C++ dasturlash tili tarkibidagi ko‘rsatkichlarga boshlang‘ich qiymatlarmi berish uchun, albatta, biror o‘zgaruvchi orqali qabul qilish maqsadga muvofiq bo‘ladi. Ko‘rsatkichlarga boshlang‘ich qiymatlarni = belgisi yoki qavs ichiga olib berilish mumkin. Ko‘rsatkichlarni boshlang‘ich qiymatlarini berish uchun quyidagi dasturga e’tibor bering:

```
int main()
{
   int n=9,m=10;
   int *y=&n;
   int *x(&m);
   cout<<x<<endl;
   cout<<y<<endl;
   return 0;
}
```

C++ dasturlash tilida ko‘rsatkichlar ustida bajariladigan amallar quyidagilarni tashkil etadi:

-ob’yektga vositali murojat qilish amali;
-qiymat berish amali;
-ko‘rsatgichga o‘zgarmas qiymatni qo‘shish amali;
-ayirish amali;
-inkrement va dekrement amali;
-solishtirish amali.

C++ dasturlash tilida ko‘rsatkichlar ustida faqatgina yuqorida keltirilgan amallardan foydalanish maqsadga muvofiq bo‘ladi.

Ko‘rsatgichni aniqlash

Ko‘rsatgichni aniqlash uchun siz ko‘rsatgich ko‘rsatadigan ob’yekt turini va yulduzcha belgisini ko‘rsatishingiz kerak:

ma’lumot_tipi* ko‘rsatkich_nomi;

Ko‘rib turganimizdek, birinchi navbatda ko‘rsatgich ko‘rsatadigan ma’lumotlar turi va yulduzcha belgisi * mavjud va undan keyin ko‘rsatgich nomi beriladi.

Misol sifatida *int* tipidagi ob’yektga ko‘rsatgichni aniqlash quyidagi ko‘rinishda bo‘ladi:

```
int* p;
```

Bunday ko‘rsatgich faqat *int* turidagi o‘zgaruvchining manzilini saqlashi mumkin, ammo hozircha bu ko‘rsatgich hech qanday ob’yektga murojaat qilmaydi va tasodifiy qiymatni saqlaydi. Biz hatto uni konsolga chiqarishga harakat qilishimiz mumkin:

```
#include <iostream>

int main()
{
   int* p;
   std::cout << p << std::endl;
}
```

Masalan, mening holatimda konsol "0x8" ni chiqardi — o‘n oltilik formatdagi ba’zi manzillar (odatda o‘n oltilik shakl xotirada manzillarni ifodalash uchun ishlatiladi). Ammo ko‘rsatgichni ba’zi qiymatlar bilan boshlash ham mumkin:

```
int* p{};
```

Muayyan qiymat ko‘rsatilmaganligi sababli, ko‘rsatgich qiymat sifatida 0 raqamini oladi. Ushbu qiymat hech narsani ko‘rsatmaydigan maxsus manzilni anglatadi. Bundan tashqari, aniq *null* bilan boshlash mumkin, masalan, maxsus *nullptr* doimiyidan foydalanish mumkin:

```
int* p{nullptr};
```

Umuman olganda hech kim ko‘rsatgichlarni ishga tushirishni taqiqlamaydi. Biroq, yuqoridagi kabi ma’lum bir qiymat yoki nol bilan boshlash tavsiya etiladi. Shunday qilib, masalan, kelajakda nol qiymat ko‘rsatgich hech qanday ob’yektga ishora qilmasligini aniqlashini bilib oldik.

Shuni ta’kidlash kerakki, yulduzcha pozitsiyasi ko‘rsatgichning ta’rifiga ta’sir qilmaydi: uni ma’lumotlar turiga yoki o‘zgaruvchining nomiga yaqinroq joylashtirish mumkin:

```
int* p1{};
int *p2{};
```

Shuni ham ta’kidlash kerakki, ko‘rsatgich qiymatining o‘lchami (saqlangan manzil) ko‘rsatgich turiga bog‘liq emas. Bu ma’lum bir platformaga bog‘liq. 32 bitli platformalarda manzillar hajmi 4 baytga, 64 bitli platformalarda esa 8 baytga teng. Masalan:

```
#include <iostream>

int main()
{
   int *pint{};
   double *pdouble{};
   std::cout << "*pint size: " << sizeof(pint) << std::endl;
   std::cout << "*pdouble size: " << sizeof(pdouble) << std::endl;
}
```

Bunday holda, har xil turdagi ikkita ko‘rsatkich aniqlanadi — int va double. Ushbu turdagi o‘zgaruvchilar har xil o‘lchamlarga ega — mos ravishda 4 va 8 bayt. Ammo ko‘rsatgich qiymatlarining o‘lchamlari bir xil bo‘ladi. Agar yuqoridagi kodni 64-bitli platformada bajarsak ikkala ko‘rsatgichning o‘lchami 8 baytga teng.

Ko‘rsatkich adresini olish va & operatori

Yordamida operatsiya & ba’zi bir ob’yektning manzilini, masalan, o‘zgaruvchining manzilini olish mumkin. Keyin bu manzil ko‘rsatgichga tayinlanishi mumkin:

```
int number {25};
int *pnumber {&number}; // pnumber ko‘rsatgichi raqam
o‘zgaruvchisining manzilini saqlaydi
```

&number ifodasi number o‘zgaruvchisining manzilini qaytaradi. Shuning uchun number o‘zgaruvchisi number o‘zgaruvchisining manzilini saqlaydi. Eng muhimi, number o‘zgaruvchisi int turiga ega va uning manzilini ko‘rsatadigan ko‘rsatgich ham int tupiga ega.

Agar biz o‘zgaruvchining manzilini konsolga ko‘rsatishga harakat qilsak, u o‘n oltilik qiymatni ifodalashini ko‘ramiz:

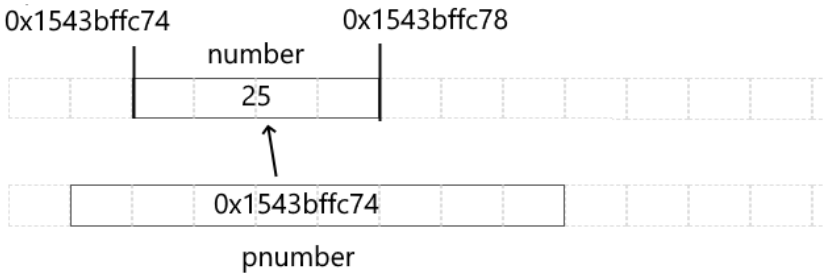
```
#include <iostream>

int main()
{
   int number {25};
   int *pnumber {&number}; // pnumber ko‘rsatgichi raqam
o‘zgaruvchisining manzilini saqlaydi
```

```
    std::cout << "number addr: " << pnumber << std::endl;
}
```

Ushbu hlatda konsolda dasturning quyidagi natija beradi:

```
number addr: 0x1543bffc74
```

Har bir alohida holatda, manzil har xil bo‘lishi mumkin va dastur turli xil ishga tushirilganda o‘zgarishi mumkin. Masalan, yuqoridagi holatda number o‘zgaruvchisining mashina manzili 0x1543bffc74. Ya’ni, kompyuter xotirasida number o‘zgaruvchisi joylashgan 0x1543bffc74 manzili mavjud. X o‘zgaruvchisi int turini ifodalaganligi sababli, aksariyat arxitekturalarda u keyingi 4 baytni egallaydi (ma’lum arxitekturalarda int turi uchun xotira hajmi farq qilishi mumkin). Shunday qilib, int turidagi o‘zgaruvchi 0x1543bffc74, 0x1543bffc75, 0x1543bffc76, 0x1543bffc77 manzillari bilan xotira hujayralarini ketma-ket egallaydi.



8.1. rasm. O‘zgaruvchining xotirada yacheykasida joylashuvi.

pnumber ko‘rsatkichi raqam o‘zgaruvchisi joylashgan manzilga, ya’ni 0x1543bffc74 manziliga murojaat qiladi.

Shunday qilib, number ko‘rsatgichi raqam o‘zgaruvchisining manzilini saqlashi ma’lum bo‘lsa, unda pnumber ko‘rsatgichining o‘zi qayerda saqlanadi? Buni bilish uchun biz pnumber o‘zgaruvchisiga & operatsiyani ham qo‘llashimiz mumkin:

```
#include <iostream>

int main()
{
    int number {25};
    int *pnumber {&number}; // указатель pnumber хранит адрес переменной number
    std::cout << "number addr: " << pnumber << std::endl;
    std::cout << "pnumber addr: " << &pnumber << std::endl;
}
```

Qiymatni manzil bo‘yicha olish

Ko‘rsatgich manzilni saqlaganligi sababli, biz ushbu manzilda saqlangan qiymatni, ya’ni number o‘zgaruvchisining qiymatini olishimiz mumkin. Buning uchun * yoki bo‘shatish operatsiyasi ("indirection operator" / "dereference operator") qo‘llaniladi. Ushbu operatsiyaning natijasi har doim ko‘rsatgich ko‘rsatadigan ob’yektdir. Biz ushbu operatsiyani qo‘llaymiz va raqam o‘zgaruvchisining qiymatini olamiz:

```
#include <iostream>

int main()
{
   int number {25};
   int *pnumber {&number};
   std::cout << "Address = " << pnumber<< std::endl;
   std::cout << "Value = " << *pnumber << std::endl;
}
```

9.1-masala. $\frac{\sin 5+1,75}{3e^{\cos 7}}$ ifodani hisoblaydigan dastur yozing.

Dastur kodi:	Dastur natijasi:
#include <iostream> #include <cstdlib> #include <cmath> **using namespace** std; **int main**() { cout<<(sin(**5.0**)+pow(**1.75**,**2.0**))/ (**3***exp(cos(**7.0**)))<<endl; **return 0**; }	0.32993

9.2-masala. Uchburchakning ikki tomonining uzunligini va ular orasidagi burchakning kattaligini va qolgan tomoni uzunligini ko‘rsatadigan dastur kodini yozing.

Dastur kodi:	Dastur natijasi:

#include <iostream>
#include <cstdlib>
#include <cmath>
using namespace std;
int main()
{
double a,b,alfa;
cout<<"ikki tomon uzunligini
kiriting: ";
cin>>a>>b;
cout<<"Ular urtasidagi burchakni
kiriting: ";
cin>>alfa;
cout<<" Qolgan tomonning uzunligi="
<<sqrt(a*a+b*b-**2***a*b*cos(alfa))<<endl;
return 0;
 	
}
	ikki tomon uzunligini kiriting: 4
	5
	Ular o‘rtasidagi burchakni kiriting: 90
	Qolgan tomonning uzunligi=7.67613

Nazorat savollari:

1. C++da qanday matematik funksiyalar mavjud?
2. C++da trigonometrik qanday trigonometrik funksiyalar mavjud?
3. C++da matematik funksiyalardan foydalanish uchun qaysi kutubxona faylini ulash kerak?

9-mavzu. C++ da matematik funksiyalar.

C++ dasturlash tilida raqamlarda matematik vazifalarni bajarish imkoniyatini beradigan ko‘pgina funksiyalar mavjud.

MIN va MAX

Ushbu funksiya yordamida argument sifatida berilgan o‘zgaruvchilarning eng yuqori qiymatini topish uchun foydalanish mumkin:

cout << max(13, 15);

va usbu oz’garuvchilarning kichik qiymatini topish uchun ishlatilishi mumkin:

cout << min(5, 10);

<cmath> Kutubxonasi. sqrt(Kvadrat ildiz), round(sonni aylantiradi) va log (natural logarifm) kabi boshqa funksiyalarni <cmath> kutubxonasida topish mumkin :

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
  cout << sqrt(64) << "\n";
  cout << round(2.6) << "\n";
  cout << log(2) << "\n";
  return 0;
}
```

Natija:

8

3

0.693147

Boshqa matematik funkstyalar. Boshqa Math funksiyalari (<cmath>kutubxonasida) ro‘yxatini quyidagi jadvalda topish mumkin:

9.1-jadval. <cmath> kutubxonasi funksiyalari va ularning vazifalari jadvali

Funksiya	Vazifasi
abs(x)	X ning mutlaq qiymatini qaytaradi. Ya’ni X sonni modulini qaytaradi.
acos(x)	X ning arccos radian bilan qaytaradi
asin(x)	X ning arcsin radian bilan qaytaradi

atan(x)	X ning arctg radian bilan qaytaradi
cbrt(x)	X ning kub ildizini qaytaradi
ceil(x)	X qiymatini eng yaqin butun songa qadar qaytaradi
cos(x)	X ning cos radian bilan qaytaradi
exp(x)	Expotensial qiymatni qaytaradi.
expm1(x)	ex -1 ushbu qiymatni qaytaradi.
fabs(x)	Haqiqiy son x ning modul qiymatini qaytaradi
fdim(x, y)	X va y o‘rtasidagi farqni qaytaradi
floor(x)	X qiymatini eng yaqin butun songa qaytaradi
hypot(x,y)	sqrt(x2 +y2) ushbu natijani qaytaradi.
fmod(x, y)	X /y ning haqiqiy qiymatning qoldiq qismini qaytaradi
pow(x, y)	x ni y darajaga ko‘taradi.
sin(x)	sin(x) ni hisoblaydi
tan(x)	tg(x) ni hisoblaydi. ctg ni hisoblash uchun ctg = 1/ tan(x); deb yoziladi.
sqrt(x)	X ni kvadrat ildizi
log(x)	X natural logorifmi

Shuni esda tutish kerakki, ushbu funksiyalarning operandlari har doim haqiqiy bo‘lishi kerak. Buning sababi shundaki, argumentlar ro‘yxatiga mos keladigan ortiqcha yuklangan funksiyalarning bir nechta nusxalari mavjud.

9.1-Listing. Ba’zi funksiyalarni qo‘llash misoli:

```
#include <iostream>
#include <math.h>

int main()
{
   std::cout << "abs(-3) = " << std::abs(-3)<< "\n";
   std::cout << "pow(-3, 2) = " << std::pow(-3, 2)<< "\n";
   std::cout << "round(-3.4) = " << std::round(-3.2)<< "\n";
   std::cout << "ceil(3.2) = " << std::ceil(3.2)<< "\n";
   std::cout << "floor(3.2) = " << std::floor(3.2)<< "\n";
   std::cout << "ceil(-3.2) = " << std::ceil(-3.2)<< "\n";
   std::cout << "floor(-3.2) = " << std::floor(-3.2)<< "\n";
```

```
}
```

Ushbu dasturni ishga tushirganda konsolda quyidagi natija chiqadi:

```
abs(-3) = 3
pow(-3, 2) = 9
round(-3.4) = -3
ceil(3.2) = 4
floor(3.2) = 3
ceil(-3.2) = -3
floor(-3.2) = -4
```

10-mavzu. Matritsa va massivlar.

Massiv – bu bir toifali, chekli qiymatlarning tartiblangan to‘plamidir. Massivlarga misol qilib matematika kursidan ma’lum bo‘lgan vektorlarni, matritsalarni ko‘rsatishimiz mumkin.

Massiv elementiga bir indeks orqali murojaat qilish mumkin bo‘lsa u **bir o‘lchamli massiv deyiladi**.

Bir o‘lchamli massivni e’lon qilish sintaksisi quyidagicha bo‘ladi.

< massiv_tipi> <massiv_nomi> [elementlar_soni]={boshlang‘ich qiymatlar};

Massivni dastur kodida e’lon qilishga misollar:

1. float a[5];
2. int m[6];
3. bool b[10];

Massiv elementlariga murojaat qilish oddiy o‘zgaruvchilarga murojaat qilishdan biroz farq qiladi. Massiv elementiga murojaat qilish uning indeksi orqali bo‘ladi.

a[1]=10; *//a massivining indeksi 1 bo‘lgan elementi 10 qiymat o‘zlashtirsin;*

cin>>a[2]; *// a massivining indeksi 2 bo‘lgan elementi kiritilsin;*

cout<<a[3]; *//a massivining indeksi 3 bo‘lgan elementi chiqarilsin;*

C++ da massiv indeksi 0 dan boshlanadi.

C++ da massiv 0-indeksli elementga o‘rnatilgan ko‘rsatkich hisoblanadi. Shuning uchun ixtiyoriy i-elementga murojaat a[i] yoki *(a+i) orqali bo‘lishi mumkin.

Massivni e’lon qilishda uning elementlariga boshlang‘ich qiymat berish mumkin va buning bir nechta usullari mavjud:

1. O‘lchami ko‘rsatilgan massivni to‘liq initsializatsiyalash;
2. O‘lchami ko‘rsatilgan massivni to‘liqmas initsializatsiyalash;
3. O‘lchami ko‘rsatilmagan massivni to‘liq initsializatsiyallash;
4. O‘lchami ko‘rsatilgan massivning barcha elementlariga boshlang‘ich qiymat 0 berish.

Massivlar har bir qiymat uchun alohida o‘zgaruvchilarni e’lon qilish o‘rniga, bir o‘zgaruvchida bir nechta qiymatlarni saqlash uchun ishlatiladi.

Massivni e’lon qilish uchun o‘zgaruvchining turini aniqlang, massiv nomini **kvadrat qavslar** ichida belgilang va u saqlashi kerak bo‘lgan elementlar sonini belgilang:

string cars[4];

Yuqorida keltirilgan misolda qiymatlarni kiritish uchun biz literal massivdan foydalanishimiz mumkin — qiymatlarni vergul bilan ajratilgan ro‘yxatga, figurali qavslar ichiga joylashtiramiz:

```
string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
```

Uchta butun sonli massivni yaratish uchun siz quyidagilarni yozishingiz mumkin:

```
int myNum[3] = {10, 20, 30};
```

Massiv elementlariga kirish.

Kvadrat qavs ichidagi indeks raqamiga murojaat qilib, massiv elementiga murojaat qilinadi.

Ushbu kod massivdagi avtomobillardagi birinchi **elementning** qiymatiga kiradi :

```
string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
cout << cars[0];
//  Volvo tanlandi
```

Eslatma: Massiv indekslari 0 dan boshlanadi: [0] birinchi element. [1] ikkinchi element va boshqalar.

Muayyan elementning qiymatini o‘zgartirish uchun indeks raqamiga murojaat qilinadi:

```
cars[0] = "Opel";
```

Misol

```
string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = "Opel";
cout << cars[0];
// Endi volvo o‘rniga Opel chiqaradi
```

Massivlar statik va **dinamik** turga bo‘linadi:

- Statik(o‘zgarmas) massiv deb, xotirada egallagan joyi o‘zgarmas, va dastur boshida e’lon qilingan dasturlarga aytiladi.
- Dinamik massiv esa dastur davomida o‘z hajmini va xotiradab egallagan joyini o‘zgartirishi mumkin.

Massivlarni ishlatishida ayrim eslamalar:

1. Massiv elementlari 0 dan boshlanadi;
2. Indeks 0 dan n-1 gacha o‘zgarishi mumkin, bu yerda n-massiv elementlarining soni;
3. Massivlarni kiritish, chiqarish va qayta ishlash uchun sikl operatorlaridan (for,while)foydalanish qulay.
4. Massivlar bilan ishlashda indeksdan chiqishi (mavjud bo‘lmagan elementga murojaat qilish) xatolik hisoblanadi.

Ikki yoki ko‘p o‘lchovli massivlar.

Ko‘p olchovli massivlarni “massivlar massivi” deb ta’riflash mumkin. Masalan, 2 o‘lchovli massivni elementlardan tuzilgan 2 o‘lchovli jadval sifatida tasavvur qilish mumkin, ularning barchasi bir xil ma’lumotlar turiga ega:

int number s[3][2],

bunday massiv 3 ta elementdan iborat bo‘lib, har bir element 2 ta elementdan iborat massivni ifodalaydi.

Ko‘p o‘lchovli massivni e’lon qilish uchun o‘zgaruvchining turini aniqlash, massiv nomini belgilash, undan keyin asosiy massivda nechta element borligini ko‘rsatadigan kvadrat qavslar, keyin esa pastki massivlarning nechta elementi borligini ko‘rsatadigan boshqa kvadrat qavslar to‘plamini belgilash orqali amalga oshiriladi:

```
string letters[2][4];
```

Oddiy massivlarda bo‘lgani kabi, siz qiymatlarni massiv harfi — figurali qavslar ichida vergul bilan ajratilgan ro‘yxat bilan kiritishingiz mumkin. Ko‘p o‘lchovli massivda massiv literalidagi har bir element boshqa massiv literalidir.

```
string letters[2][4] = {
 { "A", "B", "C", "D" },
 { "E", "F", "G", "H" }
};
```

Massiv deklaratsiyasidagi har bir kvadrat qavs to‘plami massivga boshqa **o‘lcham qo‘shadi.** Yuqoridagi kabi massiv ikki o‘lchovli deyiladi.

Massivlar har qanday miqdordagi o‘lchamlarga ega bo‘lishi mumkin. Massivning o‘lchamlari qanchalik ko‘p bo‘lsa, kod shunchalik murakkablashadi. Quyidagi massiv uch o‘lchamga ega:

```
string letters[2][2][2] = {
 {
  { "A", "B" },
  { "C", "D" }
 },
 {
  { "E", "F" },
  { "G", "H" }
 }
};
```

Ko‘p o‘lchovli massivning elementiga nurojaat qilish uchun massivning har bir o‘lchamida indeks raqamini belgilash orqali amalga

oshiriladi. Quyida string massivining birinchi qatoridagi (0) va uchinchi ustunidagi (2) element qiymatiga murojaat qlish imkonini beradi:

```
    string letters[2][4] = {
  { "A", "B", "C", "D" },
  { "E", "F", "G", "H" }
};

cout << letters[0][2];  // "C" chiqadi
```

Eslatma: massiv indekslari 0 dan boshlanadi: [0] birinchi element. [1] ikkinchi element va boshqalar.

Ko‘p o‘lchovli massivdagi elementlarini o‘zgartirish

Elementning qiymatini o‘zgartirish uchun har bir o‘lchamdagi elementning indeks raqamiga e’tibor bering:

```
    string letters[2][4] = {
  { "A", "B", "C", "D" },
  { "E", "F", "G", "H" }
};
letters[0][0] = "Z";

cout << letters[0][0];  // Hozir “A” o‘rniga “Z” chiqaradi
```

Ko‘p o‘lchovli massivda har biri elimentlaridan aylanib chiqish uchun massivning har bir o‘lchami uchun bitta sikl kerak bo‘ladi. Quyidagi misol string qatoridagi barcha elementlarni chiqaradi:

```
    string letters[2][4] = {
  { "A", "B", "C", "D" },
  { "E", "F", "G", "H" }
};

for (int i = 0; i < 2; i++) {
  for (int j = 0; j < 4; j++) {
    cout << letters[i][j] << "\n";
  }
}
```

10.1-masala. n ta elementdan tashkil topgan massiv berilgan. Massiv elementlari orasida, chap qo‘shnisidan katta bo‘lgan elementlarining indekslarini o‘sish tartibida chiqaruvchi va ularning sonini chiqaruvchi dastur tuzilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main()
{
int a[10], n;
cout << "n="; cin >> n;
for(int i = 0; i < n ; i++)
{
cout << "a[" << i << "]="; cin >> a[i];
}
for(int i = 0; i < n — 1; i++)
{
if(a[i] > a[i + 1]) cout << i << " "; } }
	N ni kiriting:
	N=5
	5 4 3 2 1
	0 1 2 3

10.2-masala. n ta elementdan tashkil topgan massiv berilgan. Massiv elementlari orasidan birinchi uchragan lokal minimum elementi indeksini chiqaruvchi dastur tuzilsin. Lokal minimum — o‘ng va chap qo‘shnisidan kichik bo‘lgan element.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main()
{
int a[10], n;
bool p = true;
cout << "n="; cin >> n;
for(int i = 0; i < n; i++)
{
cout << "a[" << i << "]="; cin >> a[i];
}
for(int i = 1; i < n — 1; i++)
{
if(a[i] < a[i — 1] && a[i] < a[i + 1])
{
cout << i;
p = false;
	N ni kiriting:
	6
	4 2 3 4 5 6
	1 ta local minimum bor.

```
        break;
      }
    }
    if( p ) cout << "Lokal minimum yo‘q";
}
```

10.3-masala. n ta elementdan tashkil topgan massiv berilgan. Massiv elementlari orasidan juftlarini indekslari kamayish tartibida chiqaruvchi va ularning sonini chiqaruvchi dastur tuzing.

Masalan:

Massiv elementlari:4 5 7 8 6 9

Natija:6 8 4 , soni=3

Dastur kodi:

```
#include<iostream>
using namespace std;
int main()
{
  int a[10], n, juftsoni = 0;
  cout << "n="; cin >> n;
  for(int i = 0; i < n; i++)
  {
    cout << "a[" << i << "]="; cin >> a[i];
  }
  for(int i = n — 1; i >= 0; i--)
  {
    if( a[i] % 2 == 0)
    {
      cout << a[i] << " ";
      juftsoni ++;
    }
  }
  cout << "Juftlar soni = " << juftsoni;
}
```

Dastur natijasi:

```
N ni kiriting:
4
2 3 4 5
 4 va 2
Juftlar soni 2 ta
```

10.4-masala. n ta elementdan tashkil topgan a va b massiv berilgan. a va b massiv qiymatlarini almashtiruvchi va ekranga oldin a massivni keyin b massivni chiqaruvchi dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main()
{
int a[10], b[10], n, k;
cout << "n="; cin >> n;
for(int i = 0; i < n; i++)
{
cout << "a[" << i << "]="; cin >> a[i];
}
for(int i = 0; i < n; i++)
{
cout << "b[" << i << "]="; cin >> b[i];
}
for(int i = 0; i < n; i++)
{
k = a[i];
a[i] = b[i];
b[i] = k;
}
for(int i = 0; i < n; i++)
{
cout << "a[" << i << "]=" << a[i] << " ";
}
cout << endl;
for(int i = 0; i < n; i++)
{
cout << "b[" << i << "]=" << b[i] << " ";
}
	N ni kiriting:
	3
	A massiv elementlarini kiriting:
	1 4 9
	B massiv elementlarini kiriting:
	2 8 7
	A massiv elementlari:
	2 8 7
	B massiv elementlari:
	1 4 9

10.5-masala. n ta elementdan iborat massiv va k butun soni berilgan (1<=k<n). Indeksi k ga teng bo‘lgan elementlarni o‘chiruvchi dastur tuzilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main()
{
int n, k;
cout << "n = "; cin >> n;
cout << "k (0 <= k < n)= "; cin >> k;
int *a = new int [n];
for(int i = 0; i < n; i++)
{
cout << "a[" << i << "]="; cin >> a[i];
}
cout << "O‘chirilishi kerak bo‘lgan element a[" << k << "]=" << a[k] << endl;
for(int i = k; i < n — 1 ; i++)
{
a[i] = a[i + 1];
 	
}
for(int i = 0; i < n — 1; i++)
{
cout << "a[" << i << "]=" << a[i] << ’\n’;
}
delete [] a;
}
	N ni kiriting:
	3
	K ni kiriting:
	2
	Massiv elementlari:
	1 4 7
	O‘chirilishi kerak bo‘lgan element:
	a[2]=7

10.6-masala. Tekislikdagi N ta nuqta koordinatalari (x,y) berilgan. Shu nuqtalarni kamayish tartibida joylashtiruvchi dastur tuzilsin. (x1,y1)<(x2,y2) hisoblanadi, agar x1+y1<x2+y2 bo‘lsa , yoki x1+y1=x2+y2 va x1<x2 bo‘lsa.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<math.h>
using namespace std;
int main()
{
int n;
cout << "n = "; cin >> n;
	N ni kiriting:
	4
	x va y elementlarini kiriting:
	1 2 3 4 6 8 7 6
	6 8
	7 6

```
float *x = new float [n], *y = new float [n];
for(int i = 0; i < n; i++)
{
  cin >> x[i] >> y[i];
}
for(int i = 0; i < n — 1; i++)
{
  for(int j = n — 1; j > i; j--)
  {
    if(x[j] + y[j] > x[j — 1] + y[j — 1])
  {
    float t = x[j];
    x[j] = x[j — 1];
    x[j — 1] = t;
    t = y[j];
    y[j] = y[j — 1];
    y[j — 1] = t;
  }
  else if(x[j] + y[j] == x[j — 1] + y[j — 1])
  {
    if(x[j] > x[j — 1])
  {
    float t = x[j];
    x[j] = x[j — 1];
    x[j — 1] = t;
    t = y[j];
    y[j] = y[j — 1];
    y[j — 1] = t;
  }
  }
}
}
for(int i = 0; i < n; i++)
cout << x[i] << " " << y[i] << endl;;
  delete [] x;
  delete [] y;
}
```

```
3 4
1 2
```

10.7-masala. n,k natural sonlari va n ta haqiqiy son berilgan. Shu sonlarning k-darajasini chiqaruvchi dastur tuzilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int n,k;
float a[100], s=1;
cout<<" n = "; cin>>n;
cout<<" k = "; cin>>k;
cout<<"sonlarni kiriting:"<<endl;
for(int i=0; i<n; i++){
cin>>a[i];
}
for(int i=0; i<n; i++){
for(int j=0; j<k; j++){
s*=a[i];}
cout<<s<<endl;
s=1;}
}
	n va k sonlarini kiriting:
	n=2 k=3
	elementlarni kiriting:
	1 4
	1 64

10.8-masala. m va n butun musbat sonlari berilgan. m x n o‘lchamli matritsani shunday hosil qilingki , uning har bir i-satri elementlari 10 * i ga teng bo‘lsin. (i=0,1,…..,n-1)

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int a[10][10],m,n;
cout<<"m="; cin>>m;
cout<<"n="; cin>>n;
for(int i=0; i<m; i++)
for(int j=0; j<n; j++)
a[i][j]=10*i;
cout<<"Matritsa elementlari : \n";
for(int i=0; i<m; i++)
	M va n sonlarini kiriting:
	2 va 4
	0 0 0 0
	10 10 10 10

```
        {
                for(int j=0; j<n; j++)
                {
                        cout<<a[i][j]<<" ";
                }
                cout<<endl;
        }
    }
```

11-mavzu. Belgi va satrlar.

Matnni saqlash uchun satrlar ishlatiladi.O‘zgaruvchi stringda ikki tirnoq bilan o‘ralgan belgilar to‘plamini o‘z ichiga oladi:

Misol:

String turidagi o‘zgaruvchilarni yarating va unga qiymat bering.

string greeting=”Hello”;

Satrlardan foydalanish uchun siz manba kodiga , kutubxona qo‘shimcha sarlavha faylini kiritishingiz kerak.

C++ da satrlar bilan ishlashni qulaylashtirish uchun string sinfi kiritilgan. Standart kutubxonadagi string sinfidan foydalanish uchun <string> sarlavha faylini dasturga qo‘shish kerak. Lekin ba’zi eski kompilyatorlarda <cstring.h> yoki <bstring.h> sarlavha faylini qo‘shish kerak bo‘ladi. Oddiy eski usuldagi satrlar bilan ishlash uchun esa , <string.h> sarlavha fayli qo‘shiladi.

Satrlar bilan ishlash , uning xususiyatlarini aniqlash uchun ba’zi bir funksiyalardan ham foydalansa bo‘ladi.

Masalan:

```
int size()          //satr o‘lchami
int length()        //satr elementlari soni
int max_size()      //satrning maximal uzunligi
int capacity()      //satr egallagan xotira hajmi
bool  empty()       //satrning bo‘shligini aniqlash
```

Satr uzunligini o‘zgartirish uchun resize funksiyasidan foydalaniladi.

void clear(); — funksiyasi satrni tozalash (to‘liq o‘chirish) uchun ishlatiladi.

bool empty() ;- funksiyasi satrni bo‘shligini tekshirish uchun ishlatiladi. Agar satr bo‘sh bo‘lsa true qiymat qaytaradi.

Foydalanuvchi kiritish satrlari. Foydalanuvchi kiritgan qatorni saqlash uchun cin >> operatoridan foydalanish mumkin.

Biroq, cin bo‘sh joyni (bo‘sh joylar, tabulyatsiya va boshqalar) tugatish belgisi sifatida ko‘rib chiqadi, ya’ni u faqat bitta so‘zni saqlashi mumkin (hatto ko‘p so‘zlarni yozsangiz ham):

string TuliqNom;

```
cout << "Tuliq ism sharifingizni kiriting: ";
cin >> TuliqNom;
cout << "Sizning ism sharifingiz: " << TuliqNom;
```

Dastur natijasi

```
// Tuliq ism sharifingizni kiriting: Sabohat Tillayeva
// Your name is: Sabohat
```

Yuqoridagi misoldan siz dastur "Sabohat Tillayeva" ni chop etishini kutgansiz, lekin u faqat "Sabohat" ni chop etadi.

Shuning uchun satrlar bilan ishlaganda qatorni o‘qish uchun ko‘pincha getline() funksiyasidan foydalanamiz. U birinchi parametr sifatida cinni, ikkinchi parametr sifatida satr o‘zgaruvchisini oladi:

```
string TuliqNom;
cout << "Ism sharifingizni to‘liq kiriting: ";
getline (cin, TuliqNom);
cout << "Sizning to‘liq ism sharifingiz: " << TuliqNom;
// Ism sharifingizni to‘liq kiriting: Bekzod Abdullayev
// Sizning to‘liq ism sharifingiz: Bekzod Abdullayev
```

Kirish satrlari. Siz satrdagi belgilarga uning kvadrat qavs ichidagi indeks raqamiga murojaat qilish orqali kirishingiz mumkin [].

Masalan, myString dagi birinchi belgini chop etish quyidagicha:

```
string myString = "Salom!";
cout << myString[0];
// Natija Salom!
```

String indekslari 0 dan boshlanadi: [0] birinchi belgi. [1] ikkinchi belgi va boshqalar. Satrdagi ma’lum bir belgining qiymatini o‘zgartirish uchun indeks raqamiga murojaat qiling va bitta tirnoqdan foydalaning:

```
string myString = "Salom!";
myString[0] = ’K’;
cout << myString;
// Hello o‘rniga Kalom chiqadi
```

C++ satrlarni birlashtirish

+ operatoridan satrlar orasiga ularni yangi satr yaratish uchun qo‘shish uchun foydalanish mumkin. Bunga **birikma** deyiladi.

Misol:

```
string firstName = "Ilhom ";
string lastName = "Farmonov";
string fullName = firstName + lastName;
cout << fullName;
```

Yuqoridagi misolda biz chiqishda Ilhom va Farmonov o‘rtasida bo‘sh joy yaratish uchun firstName dan keyin bo‘sh joy qo‘shdik. “ “ , biroq qo‘shtirnoq bilan ham bo‘sh joy qo‘shishingiz mumkin ’ ’;

Qo‘shish:

C++ tilida satr aslida satrlarda muayyan amallarni bajara oladigan funksiyalarni o‘z ichiga olgan ob’yektdir. Masalan, satrlarni funksiya bilan birlashtirish ham mumkin append():

Misol:

```
string firstName = "Ilhom ";
string lastName = "Farmonov";
string fullName = firstName .append( lastName);
cout << fullName;
```

String toifasidagi satrni char toifasiga o‘tkazish uchun c_str yoki data funksiyalaridan foydalanish mumkin.

C++ da satr deb satr oxiri (“/0”) belgisi bilan tugaydigan belgilar massiviga aytiladi. Satrni kiritishda cin.getline funksiyasidan foydalanish mumkin.

Satrlarni ta’riflashga misollar:

```
string st (“Baxo\n”); // simvollar satri bilan initsiallash.
string st2; //bo‘sh satr
satr st3 (st); //shu tipdagi o‘zgaruvchi bilan initsiallash;
```

Satrlar ustida amallar:

- Qiymat berish (=);
- Ikki amal ekvivalentlikni tekshirish uchun (==) va (!=);
- Konkatenatsiya yoki satrlarni ulash (+);
- Qiymat berib qo‘shish amali (+=);
- Indekslash ([]);

Satr uzunligini aniqlash uchun **size()** funksiyasidan foydalaniladi.

cout<<”uzunlik”<<st<<” “<<st.size();

Maxsus **empty()** usuli orqali satr bo‘sh bo‘lsa **true,** bo‘sh bo‘lmasa **false** qiymat qaytaradi.

if (st.empty()) // true, chunki satr bo‘sh.

String sinfi obyektidan **substring** usuli yordamida ostki satrni ajratib olishimiz mumkin.

- Agar bitta simvol bo‘lsa o‘sha o‘sha ko‘rsatilgan indeksdan oxirigacha hamma simvollardam nusxa olinadi.
- Birinchi va oxirgi indeksni ko‘rsatish mumkin, bu holda ko‘rsatilgan indeksdan ikkinchi indeksgacha bo‘lgan hamma simvollardan nusxa olinadi.

“Hello world!” substring (6) ->”world!”;

“Hello world!” substring (3,8) ->”lowo”;

Concat usulida satrlarni ulash. Bu usul string sinfi yangi obyektini yaratib, berilgan satrni ko‘chiradi va usul parametrida ko‘rsatilgan satrni ulaydi.

“Hello” concat “world!” -> “Hello world!”;

Simvollarni almashtirish **replace** usuliga parameter sifatida 2 simvol uzatiladi. Satr yangi nusxasida 1-simvolga almashtiriladi.

“Hello” replace (ll,ww) -> “Hewwo”;

“Hello” toLowerCase() -> ”hello”;

“hello” toUpperCase() -> “Hello”;

String sinflari. Ko‘pgina C++ kompilyatorlari turli xil amaliy muammolarni hal qilish uchun ishlatilishi mumkin bo‘lgan sinf kutubxonalarini o‘z ichiga oladi. O‘rnatilgan sinflardan biri String sinfidir.

C++ tili StrCpy () funksiyasini o‘z ichiga olgan satr oxiri nol belgisi va satr funksiyalari kutubxonasidan meros bo‘lib qolgan. Ammo bu xususiyatlarning barchasini obyektga yo‘naltirilgan dasturlashda ishlatib bo‘lmaydi. String sinfi o‘rnatilgan a’zo va a’zo o‘zgaruvchilar to‘plamini, shuningdek, foydalanuvchidan buyruqlar olish orqali matn satrlarini qayta ishlash bilan bog‘liq ko‘plab muammolarni avtomatik ravishda hal qilish imkonini beruvchi kirish usullarini taklif etadi.

Agar sizning kompilyatoringizda o‘rnatilgan String sinfi bo‘lmasa, siz o‘zingizning string sinfingizni yaratishingiz kerak bo‘ladi.

Boshqa massivlar kabi, belgilar massivlari ham statikdir. Siz e'lon qilganingizda yoki ishga tushirganingizda ularning hajmini o'zlashtirishingiz kerak.

String ga oid masalalar.

11.1-masala. Biror bir natural son berilgan.Bu son n(n>1) ta raqamdan tashkil topgan. Sonning barcha raqamlarini n-darajaga ko'tarilgandagi yig'indisi shu sonning o'ziga teng bo'lsa bunday sonlar Armstrong sonlari deyiladi. 153 va 1634 Armstrong sonlariga misol bo'lishi mumkin, chunki $153 = 1^3 + 5^3 + 3^3$ yoki $1634 = 1^4 + 6^4 + 3^4 + 4^4$. Barcha n ta raqamdan tashkil topgan Armstrong sonlarini topuvchi dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<cmath>
using namespace std;
int main() {
string s;
cin>>s;
int sum=0,i;
for(i=0;i<s.size();i++)
sum+=pow(s[i]-48,s.size());
if(to_string(sum) == s)cout<<"Armstrong son";
else cout<<"Armstrong emas";
}
	n ni kiriting:
	1634
	Armstrong son

11.2-masala. O'n oltilik sanoq sistemasidagi son berilgan. Uni o'nlik sanoq sistemasiga o'tkazuvchi dastur tuzing.

Dastur kodi:	Dastur natijasi:
#include <iostream> #include <string> #include <cmath> using namespace std; int main() { string s; cout << "s ni kiriting : "; cin >> s;	S ni kiriitng: 111001 111001 o'nlik sistemada 1118209 kabidir

```
        int d = 0;
        int b = 1;
        int len = s.length();
        for (int i = len — 1; i >= 0; i--) {
           if (s[i] >= ’0’ && s[i] <= ’9’) {
              d += (s[i] — ’0’) * b;
           } else if (s[i] >= ’A’ && s[i] <= ’F’) {
              d += (s[i] — ’A’ + 10) * b;
           } else if (s[i] >= ’a’ && s[i] <= ’f’) {
              d += (s[i] — ’a’ + 10) * b;
           }
           b *= 16;
        }
        cout <<  s << " o‘nlik sistemada  " << d
<<" kabidir"<< endl;
        return 0;
     }
```

11.3-masala. Bo‘sh bo‘lmagan s satr berilgan. s satrda joylashgan simvollarning orasiga bittadan bo‘sh joy qo‘yib chop etilsin.

Dastur kodi:	Dastur natijasi:

`#include<iostream>`
`#include<string>`
`using namespace std;`
`int main() {`
`string s;`
`cout<<"satr kiritilsin : ";`
`getline(cin,s);`
`for(int i=0; i<s.size(); i++) {`
`cout<<s[i]<<" ";  } }`
	Satrni kiriting:
	assalom
	a s s a l o m

11.4-masala. Berilgan sonni aytilgan xonagacha yaxlitlaydigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include <iostream>
#include<iomanip>
using namespace std;
int main() {
double n,x;
cout<<"sonni kiriting ";cin>>n;
cout<<"verguldan keyingi nechta xonani yaxlitlash kerak ";cin>>x;
cout<<setprecision(x)<<fixed<<n;
}
	Sonni kiriting:
	154.154447
	Verguldan keyin nechta xona yaxlitlash kerak:
	4
	154.1544

11.5-masala. c simvoli berilgan. c simvoldan (jadvalda) oldin va keyin joylashgan 2 ta simvol chop etilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<string>
using namespace std;
int main(){
char c;
cout<<"belgini kiriting : ";
cin>>c;
cout<<"bu belgidan oldin va keyin keluvchi belgilar : "<<endl;
cout<<char(int(c)-1)<<" "<<char(int(c)+1);
}
	Belgini kiriting:
	c
	bu belgidan oldin va keyin keuvchi belgilar:
	b d

11.6-Masala. Sharning hajmini topadigan dastur tuzing. Buni Monte Karlo usulida amalga oshiring.

Dastur kodi:	Dastur natijasi:
#include <iostream> #include <cmath> using namespace std; int main() { double r; cin>>r;	R ni kiriting: 25 65416.7

```
    cout<<4*3.14 *r*r*r/3;
  }
```

11.7-masala. Berilgan yil kabisa yili yoki yoki yo‘qligini ekanligini topadigan dastur tuzing.

Dastur kodi:

```
#include<iostream>
using namespace std;
int main() {
  int y,m;
  cout<<"yilni kiriting : ";
  cin>>y;

  if((y%4==0  && y%100!=0) ||
(y%400==0)){

                cout<<"kabisa yili ";
  }

  else cout<<"kabisa yili emas";}
```

Dastur natijasi:

```
y ni kiriting:
2005
Kabisa yili emas
```

11.8-masala. Kun:oy:yil ko‘rinishadagi sana berilgan. Keyingi kunni topadigan dastur tuzing.

Dastur kodi:

```
#include <iostream>
using namespace std;
int main() {
   int kun, oy, yil;
   char c;
   cout << "kun:oy:yil ko‘rinishida sanani
kiriting: ";
   cin >> kun >> c >> oy >> c >> yil;
   bool a = (yil % 4 == 0 && yil % 100 !=
0) && (yil % 400 == 0);
   bool b = true;
   if (oy < 1  && oy > 12) {
      b = false;
   } else if (kun < 1 &&  kun > 31) {
```

Dastur natijasi:

```
Kun:oy:yil
ko‘rinishidagi  sanani
kiriting:
10:9:2023
Kelasi kun :
11:9:2023
```

```
            b = false;
        } else if (oy == 4 || oy == 6 ||  oy == 9 ||
oy == 11) {
            if (kun > 30) {
                b = false;
            }
        } else if (oy == 2) {
            if (a && kun > 29) {
                b = false;
            } else if (!a && kun > 28) {
                a = false;
            }
        }
        if (b) {
            kun++;
            if (oy == 2) {
                if (a && kun > 29) {
                    kun = 1;
                    oy++;
                } else if (!a && kun > 28) {
                    kun = 1;
                    oy++;
                }
            } else if (oy == 4  || oy == 6 ||  oy ==
9 || oy == 11) {
                if (kun > 30) {
                    kun = 1;
                    oy++;
                }
            } else {
                if (kun > 31) {
                    kun = 1;
                    oy++;
                    if (oy > 12) {
                        oy = 1;
                        yil++;
                    }
```

```
            }
        }
        cout << "kelasi kun : " << kun << c
<< oy << c << yil << endl;
    } else {
        cout << "xato sana" << endl;
    }   return 0;
}
```

11.9-masala. Satr berilgan. Uni joylashish o‘rniga nisbatan teskari tartibda chop etilsin.

Dastur kodi:	Dastur natijasi:

`#include<iostream>`
`#include<string>`
`using namespace std;`
`int main(){`
`string s;`
`cout<<"satrni kiriting : ";`
`getline(cin,s);`
`for(int i=s.size(); i>=0; i--){`
`cout<<s[i];`
`}`
`}`
	Satrni kiriitng:
	Salom
	molas

11.10-masala. Kun:oy:yil ko‘rinishidagi ikkita sana berilgan. Bu sanalar orasida necha kun borligini hisoblaydigan dastur tuzing.

Dastur kodi:	Dastur natijasi:

`#include <iostream>`
`using namespace std;`
`int main() {`
`cout<<"probel bilan kiriting \n 1 01 2023 shunga oxshab \n";`
`int day1 ;`
`cin>>day1;`
`int month1;`
`cin>>month1;`
`int year1;`
`cin>>year1;`
`int day2 ;`
	Probel bilan kiriting:
	1 01 2023 shunga o‘xshab;
	
	13 08 2005
	12 10 2005
	59

```
        cin>>day2;
        int month2;
        cin>>month2;
        int year2;
        cin>>year2;
        int days = 0;
        int daysInMonth[] = {31, 28, 31, 30, 31,
30, 31, 31, 30, 31, 30, 31};
        if ((year1 % 4 == 0 && year1 % 100 !=
0) || year1 % 400 == 0) {
            daysInMonth[1] = 29;
        }
        for (int month = month1; month <= 12;
month++) {
            if (month == month1) {
                days += daysInMonth[month — 1]
— day1 + 1;
            } else {
                days += daysInMonth[month — 1];
            }
        }

        for (int year = year1 + 1; year < year2;
year++) {
            if ((year % 4 == 0 && year % 100 !=
0) || year % 400 == 0) {
                days += 366;
            } else {
                days += 365;
            }
        }
        if ((year2 % 4 == 0 && year2 % 100 !=
0) || year2 % 400 == 0) {
            daysInMonth[1] = 29;
        } else {
            daysInMonth[1] = 28;
        }
```

```
        for (int month = 1; month < month2;
month++) {
            if (month == month2 — 1) {
                days += day2;
            } else {
                days += daysInMonth[month — 1];
            }
        }
        cout<<days;
        return 0;
    }
```

11.1-masala. Satr berilgan. Berilgan satrdagi barcha bosh harflar kichik harflarga, kichik harflar bosh harflarga aylantirilsin.

Dastur kodi:

```
#include<iostream>
#include<string>
using namespace std;
int main(){
    string a,i;
    getline(cin,a);
    for(int i=0; i<a.size(); i++)
    {
    if(int(a[i])>=65 && int(a[i])<=90)
                {
                a[i]=char(int(a[i])+32);
                }
                else if(int(a[i])>=97 &&
int(a[i])<=122)
                a[i]=char(int(a[i])-32);
    }
cout<<a;}
```

Dastur natijasi:

```
Sarni kiriting:
dasturlash
DASTURLASH
```

11.12-masala. “+” belgisidan foydalangan holatda ekranga romb shaklini chiqaruvchi dastur tuzing. Rombning o‘lchamini xoxishiy tanlang.

Dastur kodi:	Dastur natijasi:

`#include <iostream>`
`using namespace std;`
`int main() {`
`  int w;`
`  cout << "romb tomonini kiriting ";`
`  cin >> w;`
`  for (int i = 0; i < w; i++) {`
`  for (int j = 0; j < w — i — 1; j++) {`
`  cout << " ";`
`  }`
`  for (int j = 0; j < 2 * i + 1; j++) {`
`  cout << "+";`
`  }`
`  cout << endl;`
`  }`
`  for (int i = w — 2; i >= 0; i--) {`
`  for (int j = 0; j < w — i — 1; j++) {`
`  cout << " ";`
`  }`
`  for (int j = 0; j < 2 * i + 1; j++) {`
`  cout << "+";`
`  }`
`  cout << endl;`
`  }`
`  return 0;}`
	Rombni tomonini kiriting:
	W=5
	+
	+++
	+++++
	+++++++
	+++++++++
	+++++++
	+++++
	+++
	+

11.13-masala. Matn va biror bir belgi berilgan. Bu matndagi kiritilgan belgidan nechta uchraganini chiqaruvchi dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<string>
using namespace std;
int main(){
string s;
char c;
int k=0;
cout<<"matnni kiriting : ";
getline(cin,s);
cout<<"belgini kiriting : ";
cin>>c;
for(int i=0; i<=s.size(); i++){
if(s[i]==c) k++;
}
cout<<"bu matnda "<<k<<" ta "<<c<<" belgisi bor";
}
	Matnni kiriting:
	Matematika
	Belgini kiriting:
	a
	bu matnda 3 ta a belgisi bor

11.14-masala. Butun musbat son berilgan. Bu sonni tasvirlovchi raqamlardan iborat simvollar chapdan o‘ngga qaragan tartibda chop etilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<string>
using namespace std;
int main(){
string s;
cout<<"s ni kiritng : ";
cin>>s;
for(int i=s.size(); i>=0; i--){
cout<<s[i];
}
}
	Satrni kiriting:
	amaliyot
	toyilama

11.15-masala. Berilgan raqamni arab raqamlaridan rim raqamlariga aylantiruvchi dastur tuzing.

Dastur kodi:	Dastur natijasi:

#include <iostream>
#include <string>
std::string convertToRoman(int number) {
if (number <= 0 || number > 3999) {
return "xato son";
}
std::string romanNumeral;
int arabicNumeral[] = {1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};
std::string romanSymbol[] = {"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
int i = 0;
while (number > 0) {
int count = number / arabicNumeral[i];
for (int j = 0; j < count; j++) {
romanNumeral += romanSymbol[i];
number -= arabicNumeral[i];
}
i++;
}
return romanNumeral;
}int main() {
int number;
std::cout << "(1-3999): ";
std::cin >> number;
std::string romanNumeral = convertToRoman(number);
std::cout << romanNumeral << std::endl; return 0;
}
	(1-3999) oralig‘idagi sonni kiriting:
	547
	DXLVII

11.16-masala. Bo‘sh bo‘lmagan s satr va n(n>0) butun soni berilgan. s satrdagi simvollar orasiga n tadan “*” qo‘yib chop etilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
string s;
int n;
cout<<"satrni kiriting : ";
getline(cin,s);
cout<<"n ni kiriting : ";
cin>>n;
for(int i=0; i<s.size(); i++){
cout<<s[i];
for(int i=0; i<n; i++)
cout<<’*’;
}
}
	Satrni kiriting:
	Funksiya
	Sonni kiriting:
	1
	F*u*n*k*s*i*y*a

11.17-masala. Bo‘sh bo‘lmagan satr berilgan. Uning birinchi va oxirgi simvollarining kodlari chop etilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<string>
using namespace std;
int main(){
string s;
cout<<"satrni kiriting : ";
getline(cin,s);
cout<<int(s[0])<<" "<<int(s[s.size()-1]);}
	Satrni kiriting:
	dunyo
	100 111

11.18-masala. Satr va N natural soni berilgan. Shu satr belgilari orasiga N tadan ’*’ belgisi qo‘yilgan satr hosil qiluvchi va ekranga chiqaruvchi dastur tuzilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;int main(){
string s;
int n;
getline(cin,s);
cin>>n;
for(int i=0; i<s.size(); i++){
cout<<s[i];
for(int i=0; i<n; i++)
cout<<’*’;
} }
	Satrni kiriting
	Dasturlash
	Sonni kiriting:
	2
	D**a**s**t**u**r**l**a**s**h**

11.19-masala. Satr berilgan . Satrdagi hamma katta harflarni kichigiga, kichiklarini kattasiga almashtiruvchi programma tuzilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<string>
using namespace std;
int main(){
string a,i;
getline(cin,a);
for(int i=0; i<a.size(); i++)
{
if(int(a[i])>=65 && int(a[i])<=90)
{
a[i]=char(int(a[i])+32);
}
else if(int(a[i])>=97 && int(a[i])<=122)
 	
a[i]=char(int(a[i])-32);
	Satrni kiriting:
	AmalIYot
	aMALiyOT

```
    }
cout<<a;}
```

11.20-masala. N natural soni va satr berilgan. Uzunligi N ga teng bo‘lgan satrni shunday hosil qilingki, agar kiritilgan satrning uzunligi N dan katta bo‘lsa , satrning dastlabki belgilarini tashlab yuboring. Agar kiritilgan satrning uzunligi N dan kichik bo‘lsa , yangi satr boshiga nuqtalarni ’.’ qo‘shing.

Dastur kodi:	Dastur natijasi:

`#include<iostream>`
`using namespace std;`
`int main(){`
`int n; cout<<"n=";`
`cin>>n;`
`string s;cout<<"s=";`
`cin>>s;`
`if(n>s.size())`
`{`
`string b(n-s.size(),'.');`
`s=b+s;`
`}`
`else`
`{`
`string b(s,n);`
`s=b; } cout<<s;}`
	N ni kiriting:
	5
	Satrni kiriting:
	Amerika
	ka

11.21-masala. Probel bilan ajratilgan o‘zbekcha so‘zlardan iborat satr berilgan. Shu satrdagi so‘zlarni quyidagicha o‘zgartiring. Har bir so‘zning oxirgi harfi bilan bir xil bo‘lgan harflarni “.” bilan almashtiruvchi dastur tuzilsin. Masalan: ”minimum”

So‘zi “mini.u.” bo‘lib o‘zgartiriladi. Probellar soni o‘zgarishsiz qolsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int n; cout<<"n=";
cin>>n;
string s;cout<<"s=";
cin>>s;
if(n>s.size())
{
string b(n-s.size(),’.’);
s=b+s;
}
else
{
string b(s,n);
s=b;
}
cout<<s;
}
	Satrni kiriting:
	Minimum
	Mini.u.

11.22-Masala. Raqam va kichik lotin harflaridan iborat satr berilgan. Agar satrdagi harflar alifbo tartibida bo‘lsa 0 chiqaruvchi, aks holda qonuniyatni buzgan birinchi belgini chiqaruvchi dastur tuzilsin.

Dastur kodi:	Dastur natijasi:
#include<iostream> #include<string> using namespace std; int main(){ string s, harflar=" ";	Satrni kiriting: Maximum a

```
        cout<<"satrni kiriting:";
        getline(cin,s);
        for(int i=0; i<s.size(); i++)
        {
                if(int(s[i])>=97 &&
int(s[i])<=127) harflar+=s[i];
        }
        bool osadi = true;
        for(int i=0; i<harflar.size()-1; i++)
        {
                if(harflar[i]>harflar[i+1])
                {
                        osadi=false;
                        cout<<harflar[i+1];
                        break;
                }
        }
        if(osadi)
        cout<<0;
    }
```

11.23-masala. Nol soni bilan tugovchi butun sonlar to‘plami berilgan. Barcha musbat juft sonlar yig‘indisini chiqaruvchi dastur tuzilsin. Agar bunday son bo‘lmasa 0 chiqarilsin.

Dastur kodi:	Dastur natijasi:

`#include<iostream>`
`using namespace std;`
`int main(){`
`  int i,s=0,a;`
`  cout<<"sonlarni kiriting:";`
`  for( ; ; ){`
`  cin>>a;`
`  if(a>0 && a%2==0)`
`  s=s+2;`
`  if(a==0) break;`
`  }`
`cout<<s;`
`}`
	Sonlarni kiriting:
	1 2 4 5 0
	Yig‘indisi 6 ga teng

11.24-masala. n natural soni va n ta butun son berilgan. Bu sonlar orasida kamida 2 ta nol bor. Birinchi va oxirgi nol orasidagi sonlar yig‘indisini chiqaruvchi dastur tuzilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
int main(){
int n,a[100],sum=0,z,e,c=0;
cin>>n;
for(int i=0; i<n; i++){
cin>>a[i];
if(a[i]==0) {
c++;
if(c==1) z=i;
e=i;
}
}
for(int i=z; i<e; i++){
sum+=a[i];
}
cout<<sum;
}
	N ni kiriting:
	5
	Sonlarni kiriting:
	1 2 0 3 0
	3

11.25-masala. n(2<=n<=100) ta elementdan iborat butun sonli massiv berilgan. Massivning ikkinchi eng katta elementini aniqlang.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<algorithm>
using namespace std;
int main(){
int n;
cin>>n;
int a[100];
for(int i=0; i<n; i++) cin>>a[i];
sort(a,a+n);
cout<<a[n-2];
}
	N ni kiriting:
	4
	Sonlarni kiriting:
	1 2 3 4
	3

12-mavzu. Standart kutubxona funksiyalari.

ANSI C++ standartiga qo‘shimchalardan biri bu dasturchilar tomonidan ko‘plab kutubxonalar bilan ishlashda nomlarning to‘qnashuvidan qochish uchun nom bo‘sh joylaridan foydalanish qobiliyatidir.

ANSI standartlari ushbu muammoni nomlar maydoni (namespaces) yordamida hal qilish usulini taklif qiladi. Biroq, ehtiyot bo‘lish kerak, chunki barcha kompilyatorlar ushbu vositani qo‘llab-quvvatlamaydi.

Nomlar maydoni global nom bo‘sh joylarini ajratish uchun ishlatiladi, bu nom to‘qnashuvlarini yo‘q qilish yoki hech bo‘lmaganda kamaytirish uchun ishlatiladi.

Shunga ko‘ra, funksiyalar nomlar ichida ham, tashqarisida ham e’lon qilinishi mumkin. Ikkinchi holda, bunday funksiyani chaqirganda, tegishli nom maydoni aniq ko‘rsatilishi kerak.

Funksiyalar va sinflar nomi bo‘yicha chaqirish. Dastur kodini tahlil qilish va funksiyalar va o‘zgaruvchilar nomlari Listingini yaratish jarayonida kompilyator dasturni nom ziddiyatlari uchun tekshiradi. Kompilyatorning o‘zi hal qila olmaydigan ziddiyatlarni bog‘lovchi hal qilishi mumkin.

Agar siz bir xil nomdagi va bir-biriga o‘xshash ko‘rinadigan identifikatorlarni tasvirlashga harakat qilsangiz, xato xabar paydo bo‘ladi. Xato xabarni quyidagi dastur kodini kompilyatsiya qilish va ulashga urinayotganda bog‘lovchidan kelib chiqadi:

```
int integerValue = 0 ;
int main( ) {
int integerValue = 0
//….
return 0 ;}
// ikkinchi fayl.cpp
int integerValue = 0 ;
// ikkinchi oxiri.cpp
```

Kompilyatordan bunday eslatma ham bo‘lishi mumkin: identifier hiding (ID yashiringan). Uning mohiyati birinchi faylda.cpp integerValue o‘zgaruvchisini main () funksiyasida e’lon qilish xuddi shu nomdagi global o‘zgaruvchini yashiradi.

Main () funksiyasida main () dan tashqarida e'lon qilingan integervaalue global o'zgaruvchisidan foydalanish uchun ko'rish operatori (::) yordamida ushbu o'zgaruvchining globalligini aniq ko'rsatish kerak. Shunday qilib, quyidagi misolda 10 qiymati main()ichida e'lon qilingan bir xil nomdagi o'zgaruvchiga emas, balki integervalue global o'zgaruvchisiga beriladi:

```
int integerValue = 0 ;
int main( ){
int integerValue = 0 ;
:: integerValue=10; // global o'zgaruvchisini o'zlashtirish
return 0 ;}
// ikkinchi fayl.cpp
int integerValue = 0 ;
// ikkinchi oxiri.cpp
```

O'zgaruvchi, sinf yoki funksiya bo'lishi mumkin bo'lgan obyektning ko'rinishi ostida ushbu obyekt ishlatilishi mumkin bo'lgan dasturning qismi tushuniladi. Masalan, har qanday funksiyadan tashqarida e'lon qilingan va aniqlangan o'zgaruvchi fayl yoki global miqyosga ega. Uning ko'rinishi e'lon nuqtasidan faylning oxirigacha tarqaladi. Modulli yoki lokal ko'lamga ega bo'lgan o'zgaruvchi dasturiy modul ichida e'lon qilinadi. Ko'pincha lokal o'zgaruvchilar funksiya tanasida e'lon qilinadi. Quyida turli xil ko'rish sohalariga ega bo'lgan obyektlarning namunalari keltirilgan:

```
int globalScopeInt = 5 ;
void f( )
{ int localScopeInt = 10 ; }
int main( )
{ int localScopeInt = 15 ;
{ int anotherLocal = 20 ;
int localScopeInt = 30 ;}
return 0 ;}
```

globalScopeInt global o'zgaruvchisi f () va main () funksiyalarida ko'rinadi. f () funksiyasining tanasida localScopeInt o'zgaruvchisi prototipi mavjud. U esa lokal, ya'ni funksiya prototipini o'z ichiga olgan modul chegaralari bilan cheklangan. Main() funksiyasi f () funksiyasining localScopeInt o'zgaruvchisiga kira olmaydi.

F() funksiyasini bajarish tugagandan so'ng, localScopeInt o'zgaruvchisi kompyuter xotirasidan o'chiriladi, localScopeInt deb

nomlangan uchinchi o‘zgaruvchining prototipi main () funksiyasining tanasida joylashgan. U ham lokal o‘zgaruvchidir.

E’tibor bering: main() funksiyasining localScopeInt o‘zgaruvchisi f() funksiyasining bir xil o‘zgaruvchisiga zid kelmaydi. Quyidagi ikkita o‘zgaruvchining ko‘rinishi-anotherLocal va localScopeInt — modul maydoni bilan ham cheklangan. Boshqacha qilib aytganda, ushbu o‘zgaruvchilar e’lon qilingan joydan ushbu funksiya e’lon qilingan modul tanasini cheklaydigan yopiq figurali qavsgacha ko‘rinadi.

Ismlar ichki yoki tashqi aloqaga ega bo‘lishi mumkin. Ushbu ikkala atama ham bir nechta yoki bitta dasturiy ta’minot birligida ismning ishlatilishi yoki mavjudligini anglatadi. Tashqi aloqaga ega bo‘lgan har qanday nomga faqat uni belgilaydigan birlik doirasida murojaat qilish mumkin.

Masalan, ichki aloqaga ega bo‘lgan o‘zgaruvchidan faqat ushbu o‘zgaruvchi e’lon qilingan dastur blokidagi funksiyalar foydalanishi mumkin. Tashqi aloqalarga ega nomlar boshqa bloklardan xususiyatlarga ega. Ichki va tashqi aloqalarga misollar quyidagi kodni ko‘rsatadi.

```
// fayl: birinchi.cpp
int externalInt = 5 ;
const int j = 10 ;
int main()
return 0 ;
// fayl: ikkinchi.cpp
extern int externalInt ;
int anExternalInt = 10 ;
const int j = 10 ;
```

Birinchi .cpp faylda e’lon qilingan externalInt o‘zgaruvchisi tashqi aloqaga ega. Birinchi faylda e’lon qilinganiga qaramay .cpp ushbu o‘zgaruvchiga i.cpp kkinchi fayldan kirish mumkin. Ikkala faylda ham sukut bo‘yicha ichki aloqalarga ega bo‘lgan j konstantalari mavjud. Standart konstantalarning ichki aloqasini o‘zgartirish uchun siz quyidagi misolda bo‘lgani kabi ularning globalligini aniq ko‘rsatishingiz kerak:

```
extern const int j = 10 ;
// fayl: ikkinchi.cpp
extern const int j ;
#include <iostream>
```

```
int main(){
std::cout << "j = " << j << std::endl ;
return 0;}
```

Coutdan oldin std nom maydoni belgisidan foydalanishga e'tibor bering, bu sizga standart ANSI kutubxonasining barcha obyektlariga murojaat qilish imkonini beradi. Ushbu kod bajarilgandan so‘ng ekranda qator paydo bo‘ladi:

```
j = 10;
```

Standartlashtirish qo‘mitasi quyidagi misolda bo‘lgani kabi tashqi o‘zgaruvchining ko‘lamini cheklash uchun statikdan foydalanishni tavsiya etmaydi:

```
static int staticInt = 10 ;
int main()
{  //…
}
```

Agar hozirda statikdan bunday foydalanish shunchaki tavsiya etilmasa, kelajakda bunday ifodani umuman noto‘g‘ri deb hisoblash mumkin. Shuning uchun, endi statik o‘rniga nomlar maydonidan foydalanish yaxshiroqdir.

Nomlar maydonini yaratish. Ism maydonini e'lon qilish sintaksisi tuzilmalar va sinflarni e'lon qilish sintaksisiga o‘xshaydi. Kalit so‘zdan keyin pamesras mavjud bo‘lmasligi mumkin bo‘lgan nom maydonining nomi, so‘ngra ochiladigan figurali qavs. Ism maydoni ifoda oxirida nuqta-vergulsiz yopiladigan figurali qavs bilan yakunlanadi. Masalan,:

```
namespace Window
{
void move( int x, int y) ;}
```

Window nomi nomlar maydonini aniqlaydi. Bitta fayl ichida yoki turli xil tarjima birliklarida joylashgan nomlangan nom maydonlarining ko‘plab nusxalarini yaratish mumkin. Bunga C++standart kutubxonasining std nom maydoni misol bo‘la oladi. Uning ishlatilishi bu holatda standart kutubxona mantiqiy jihatdan yagona funksiyalar guruhi ekanligi bilan asoslanadi.

Ism maydonlarining asosiy maqsadi dasturning nomlangan hududida bog‘langan elementlarni guruhlashdir. Quyida bir nechta sarlavha fayllarini birlashtirgan nom maydonining misoli keltirilgan:

```
namespace Window
void move( int x, int y) ;
// header2. h
namespace Window
void resize( int x, int y ) ;
```

Nomlar maydoni ichida turlari va funksiyalarini e’lon qilish va aniqlash mumkin. C++da dasturlashning strategik yondashuvlarini muhokama qilmasdan qilolmaysiz. Dastur tuzilishining to‘g‘riligi dastur interfeysi uning protsessual qismidan qanchalik aniq ajratilganligi bilan belgilanadi. Ushbu prinsipga nafaqat sinflar bilan ishlashda, balki nom maydonlarini yaratishda ham amal qilish kerak. Quyida yana bir misol keltirilgan:

```
namespace Window {
// .boshqa prototiplar va o‘zgaruvchan ta’riflar.
void move (int x, int y); / / e’lonlar
void resize( int x, int y ) ;
// . . . boshqa prototiplar va o‘zgaruvchan ta’riflar.
void move( int x, int y )
if( x < MAX_SCREEN_X && x > 0 )
if( y < MAX_SCREEN_Y && y > 0 )
platform.move (x. ) // maxsus dastur
void resize( int x, int y )
if( x < MAX_SIZE_X && x > 0 )
if( y < MAX_SIZE_Y && y > 0 )
platform.resize (x, y ) ; // maxsus dastur
```

Ismlar maydoni qanchalik tez tartibli va tartibsiz bo‘lishini aniq ko‘rish mumkin! Bundan tashqari, ushbu misolda nomlar maydonini e’lon qilish atigi 20 satrni tashkil etadi.

13-mavzu. C++ da Funksiyalar.

Funksiya bu muayyan operatsiyalarni bajaradigan kod bloklardir.

Funksiya dasturning ma'lum koodlar ketma ketligini ajratib ko'rsatib uni nomlash imkonini beradi.

Funksiya kod bloki bo'lib, u faqat chaqirilgandagina ishlaydi. Siz parametrlar sifatida ma'lum bo'lgan ma'lumotlarni funksiyaga o'tkazishingiz mumkin.

Funksiyalar muayyan harakatlarni bajarish uchun ishlatiladi va ular kodni qayta ishlatish uchun muhimdir: Kodni bir marta aniqlaydi va uni ko'p marta ishlatadi.

Siz main () funksiyasi bilan tanishgansiz. Main () funksiyasi dastur tomonidan emas, balki operatsion tizim tomonidan chaqiriladi.

Dastur biror bir funksiya chaqiruvi kelgunga qadar satrma-satr bajariladi. Keyin boshqaruv ushbu funksiya satrlariga o'tkaziladi. Funksiya bajarilgandan so'ng darhol funksiya chaqiruvidan keyingi dastur qatoriga qaytadi.

Dasturning funksiyasi bilan quyidagi o'xshashlik mavjud. Misol uchun, agar chizish paytida qalamingiz singan bo'lsa, siz chizishni to'xtatasiz va qalam uchini yunasiz. Shundan so'ng, siz qalam singandagi chizma joyiga qaytasiz. Dastur ba'zi bir amallarini bajarishi kerak bo'lganda, ushbu amalni bajarish uchun mas'ul bo'lgan funksiya chaqiriladi, shundan so'ng dastur funksiya chaqirilgan joydan o'z ishini davom ettiradi(keying misolda ko'rish mumkin).

```
#include <iostream>
using namespcase std;
 // DemonstrationFunction Funksiyasi
// ma'lumotli xabarni ko'rsatadi
void DemonstrationFunction()
{cout << "In DemonstrationFunction\n";
}
// Main funksiyasi xabarni chiqaradi, keyin
// DemonstrationFunction funksiyasini chaqiradi va
// ikkinchi xabarni ko'rsatadi.
int main()
{
cout << "In main\n" ;
DemonstrationFunction();
```

```
cout << " Back in main\n";
return 0;
}
```

Natija:

```
In main
In DemonstrationFunction
Back in main
```

Tahlil:

5-7-qatorlarda DemonstrationFunction() funksiyasi aniqlanadi. U xabarni ko‘rsatadi va boshqaruvni dasturga qaytaradi. Main () funksiyasi 12-qatorda boshlanadi va 13-qatorda dastur boshqaruvi hozirda main () funksiyasida ekanligi to‘g‘risida xabar paydo bo‘ladi. Ushbu xabar chiqarilgandan so‘ng DemonstrationFunction () funksiyasi chaqiriladi. Ushbu murojaat natijasida DemonstrationFunction () funksiyasida joylashgan buyruqlar bajariladi. Bunday holda, butun funksiya 6-qatorda joylashgan bitta buyruqdan iborat bo‘lib, u boshqa xabarni chiqaradi. DemonstrationFunction() (7-qator) funksiyasi tugagandan so‘ng, dasturni boshqarish ushbu funksiya chaqirilgan joyga qaytariladi. Bunday holda, dasturning bajarilishi 15-qatordan davom etadi, unda main() funksiyasi ekranda yakuniy xabarni ko‘rsatadi.

Dasturda funksiyalardan foydalanishdan avval funksiyani e’lon qilishni va keyin aniqlashni talab qiladi. Funksiyani e’lon qilish orqali kompilyatorga uning nomi, qaytish tipi va parametrlari haqida xabar beriladi. Funksiyani aniqlash orqali kompilyator funksiyaning qanday ishlashini bilib oladi. Agar u oldindan e’lon qilinmagan bo‘lsa, dasturda biror bir funksiyani chaqirish mumkin emas. Funksiya prototipi deb, dasturga parametrlar soni va turi, shuningdek funksiya qaytadigan qiymat tipi to‘g‘risida ma’lumot beradigan funksiya e’loniga aytiladi.

Funksiyani e’lon qilishning uchta usuli mavjud.

- ✓ Funksiyaning prototipini faylga yozing va keyin uni dasturingizga kiritish uchun #include ifodasidan foydalaning.
- ✓ Funksiya prototipini ushbu funksiyadan foydalanadigan faylga yozing.
- ✓ Boshqa har qanday funksiya chaqirilishidan oldin funksiyani aniqlang.

Bunday holda, funksiyani aniqlash bir vaqtning o‘zida uni e’lon qiladi.

Funksiyani ishlatishdan oldin darhol aniqlash va shu bilan funksiya prototipini yaratish uchta sababga ko‘ra yaxshi deb hisoblanmaydi.

Birinchidan, faylda funksiyalarni ma’lum tartibda joylashtirish talabi dasturni foydalanish shartlarini o‘zgartirish jarayonida qo‘llab-quvvatlashni qiyinlashtiradi.

Ikkinchidan, a() funksiyalari b() funksiyasini chaqirishi vaziyati bo‘lishi mumkin, ammo ba’zi holatlarda c() funksiyalari a() funksiyasini chaqirishi kerak bo‘lishi mumkin. Shu bilan birga, b() funksiyasini aniqlashdan oldin a() funksiyasini va shu bilan birga a() funksiyasini aniqlashdan oldin b () funksiyasini aniqlash mumkin emas, ya’ni ushbu funksiyalardan kamida bittasi oldindan e’lon qilinishi kerak.

Uchinchidan, funksiya prototipiga ko‘ra, funksiya ma’lum bir parametrlar to‘plamini qabul qilishi yoki ma’lum bir turdagi qiymatni qaytarishi e’lon qilingan bo‘lsa, u holda kompilyator ushbu xatoni dasturni kompilyatsiya qilish bosqichida sezadi, bu esa uni amalga oshirish jarayonida kutilmagan hodisalardan qochadi.

Ko‘pgina o‘rnatilgan funksiyalarning prototiplari allaqachon #include yordamida dasturga qo‘shilgan sarlavha fayllariga yozilgan. Foydalanuvchilar tomonidan yaratilgan funksiyalar uchun dasturchining o‘zi dasturga tegishli prototiplarni yaratishi kerak.

Funksiya prototipi nuqta-vergul bilan tugaydigan ifoda bo‘lib, funksiya sarlavhasi, qaytish qiymati tipidan iborat. Funksiya sarlavhasi uning nomi va parametrlar ro‘yxatini anglatadi.

Prototipda va funksiyani aniqlashda qaytish tipi va sarlavhasi mos kelishi kerak. Agar bunday moslik bo‘lmasa, kompilyator xato xabarini ko‘rsatadi. Biroq, funksiya prototipida parametr nomlari bo‘lishi shart emas, u faqat ularning turlarini ko‘rsatish bilan cheklanishi mumkin. Masalan, quyidagi prototip xato emas:

```
long Area(int, int);
```

Ushbu prototip long int tipidagi qiymatni qaytaradigan va ikkita butun parametrlarini qabul qiladigan Area () deb nomlangan funksiyani e’lon qiladi. Prototipning bunday yozuvi juda maqbul bo‘lsada, bu eng yaxshi variant emas. Parametr nomlarini qo‘shish prototipingizni yanada aniqroq qiladi. Xuddi shu funksiya parametrlar nomlari bilan aniqroq ko‘rinadi:

```
long Area(int length, int width);
```

E’tibor bering, har bir funksiya uchun qaytish tipi har doim ma’lum. Agar u aniq e’lon qilinmasa, int tipi sukut bo‘yicha qabul qilinadi. Ammo, agar har bir funksiya, shu gapdan main () uchun qaytish tipi aniq e’lon qilinsa, sizning dasturlaringiz aniqroq bo‘ladi. Listing 10.3 da Area () funksiyasining prototipini o‘z ichiga olgan dastur mavjud.

Funksiya yaratish: C++ oldindan belgilangan ba’zi funksiyalarni ta’minlaydi, masalan main(), kodni bajarish uchun ishlatiladi. Lekin siz ma’lum harakatlarni bajarish uchun o‘zingizning funksiyalaringizni ham yaratishingiz mumkin.

Funksiyani yaratish (ko‘pincha e’lon qilish deb ataladi) uchun funksiya nomini, keyin qavslar **()** bilan belgilang:

Sintaksis:

```
void myFunction () {
// bajariladigan kod
}
```

Misol tushuntirilishi:

- myFunction() funksiyaning nomi;
- void funksiyasining qaytish qiymati yo‘qligini bildiradi. Qaytish qiymatlari haqida keyingi bobda keyinroq bilib olamiz;
- Funksiya tana ichida funksiya nima qilishi kerakligini belgilaydigan kod qo‘shiladi.

Funksiyani chaqirish. E’lon qilingan funksiyalar darhol bajarilmaydi. Ular “keyinroq foydalanish uchun saqlanadi” va keyinroq chaqirilganda bajariladi.

Funksiyani chaqirish uchun funksiya nomidan keyin ikkita qavs () va nuqtali vergul yoziladi;

C++ dasturlash tilida ***funksiyalar*** o‘z o‘zini chaqirish imkoniyatiga ega. Bunday funksiyalar ***rekursiyali*** (o‘z o‘zini chaqiruvchi) funksiya deyiladi.

Rekursiyali funksiyalarga qo‘yiladigan asosiy talab, qandaydir qiymatda rekursiya yolg‘on yoki rost qiymat qabul qilishi kerak. Shundagina chaqirilgan funksiyalar qaytadi. Aks holda funksiya o‘z-o‘zini davomli ravishda chaqiradi va xatolik sodir bo‘ladi.

Rekursiv funksiya, albatta return iborasidan foydalanadigan va ushbu funksiyaga qolgan murojaatlarning bajarilishini birlashtiriladigan asosiy variantga ega bo‘lishi kerak.

```
int fibonachchi (int value)
{
if (value<=0)
return 0;
else if (value==1)
return 1;
else fibonachchi (value-1)+fibonachchi;
}
```

Rekursiya — bu funksiya chaqiruvining o‘zini bajarish texnikasi. Ushbu usul murakkab muammolarni hal qilish, osonroq bo‘lgan oddiy muammolarga ajratish yo‘lini beradi.

Rekursiyani tushunish biroz qiyin bo‘lishi mumkin. Uning qanday ishlashini aniqlashning eng yaxshi yo‘li u bilan tajriba o‘tkazishdir.

Misol: Ikki raqamni bir-biriga qo‘shish oson, lekin bir qator raqamlarni qo‘shish ancha murakkab. Quyidagi misolda, rekursiya bir qator raqamlarni ikkita raqamni qo‘shish kabi oddiy vazifaga bo‘lish orqali qo‘shish uchun ishlatiladi:

Misol

```
int sum(int k) {
  if (k > 0) {
    return k + sum(k —1);
  } else {
    return 0;
  }
}

int main() {
  int result = sum(10);
  cout << result;
  return 0;
}
```

Misol tushuntirilishi:

Funksiya chaqirilganda, u barcha kichik raqamlar yig‘indisiga sum()parametr qo‘shadi va natijani qaytaradi. K 0 ga aylanganda, funksiya faqat 0 ni qaytaradi. Funksiya 0 bo‘lganda o‘zini chaqirmaganligi sababli k, dastur shu erda to‘xtaydi va natijani qaytaradi.

Eslatma: Ishlab chiquvchi rekursiya bilan juda ehtiyot bo‘lishi kerak, chunki hech qachon tugamaydigan yoki ortiqcha xotira yoki protsessor quvvatini ishlatadigan funksiyani yozish juda oson bo‘lishi mumkin. Biroq, to‘g‘ri yozilsa, rekursiya dasturlashda juda samarali va matematik jihatdan oqlangan yondashuv bo‘lishi mumkin.

Funksiya obyektlarining o‘zlarini emas, balki obyektlar qiymatlarining nusxalarini boshqaradi.

Parametrlarni havola orqali uzatishda obyektga havola uzatiladi , bu orqali biz uning qiymatini emas, balki obyektlarning o‘zini boshqarishimiz mumkin.

void square (int &a, int &b)

va funksiya prototipida ham havola qoyishimiz shart.

Havola orqali uzatishda funksiyalarning qiymatida nusxalash sodir bo‘lmaydi va funksiya obyektning o‘zidan emas, balki uning qiymatidan foydalanadi.

1. Argumentlarni havola orqali uzatish.
2. Havolalarni argument sifatida uzatish.

int &aRef=a;

int &bRef=b;

square (aRef,bRef)

Parametrlar va argumentlar.

Ma’lumot funksiyalarga parametr sifatida uzatilishi mumkin. Parametrlar funksiya ichidagi o‘zgaruvchilar vazifasini bajaradi.

Parametrlar funksiya nomidan keyin qavslar ichida ko‘rsatiladi. Siz xohlagancha ko‘p parametr qo‘shishingiz mumkin, ularni vergul bilan ajratib qo‘yasiz:

Sintaksis:

```
void functionName(parametr1 , parametr2 , parametr3){
//bajariladigan kod
}
```

Funksiyaga **parametr** uzatilsa, u **argument** deb ataladi.

- Parametrlarni qiymat bo‘yicha uzatish kichik obyektlarni funksiyaga o‘tkazish uchun ko‘proq mos keladi.
- Parametrlarni havola orqali uzatish katta obyektlarni funksiyaga o‘tkazish uchun ko‘proq mos keladi.

```
void square (const int &a, const int &b);
void square (const int &, const int &);
```

Agar funksiya massivni parameter sifatida qabul qilsa, aslida ko‘rsatgich massivning birinchi elementiga ushbu funksiyaga o‘tkaziladi:

void print (int [])

void print (int number [])

Funksiya nomlarini e’lon qilishga misollar:

void error();

void func (int,int);

int func-1(void);

float f-3 (int a, int b, int c).

Funksiya sintaksisini quyidagiha tavsiflash mumkin:

Funksiya_qaytaradigan_tip, funksiya nomi(parametrlar)

{funksiya tanasi

}

Funksiyani chaqirish u aniqlangandan keyin sodir bo‘lishi kerak.

C++ dasturlash tilida funksiyaga murojaat qilishdan oldin, uni e’lon qilgan bo‘lishimiz muhim.

Funksiyani e’lon qilish – **funksiya prototipi** deyiladi.

Funksiya parametrlari qavs ichida tip, parameter_nomi ko‘rinishida bo‘ladi.

Parametr chaqirilganda aniq qiymat uzatilmasa, funksiya standart argumentni qabul qilishi mumkin.

O‘zgarmas parametrlar:

Parameter doimiy bo‘lishi mumkin, bunday parametrlar qiymatlari o‘zgarmaydi. Funksiya ma’lum qiymatini uzatish uchun **return** operatoridan foydalanamiz.

Funksiya deklaratsiyasi va ta’rifi.

C++ funksiyasi ikki qismdan iborat:

- **Deklaratsiya:**qaytarish turi,funksiya nomi va parametrlar(agar mavjud bo‘lsa)
- **Ta’rif:**funksiyaning tanasi(bajarilish kerak bo‘lgan kod)

*void **myFunction()** { //deklaratsiya*

//funksiyaning tana qismi (ta’rif)

}

12.1-masala. Maxrajlari N dan oshmaydigan , 0 dan 1 gacha bo‘lgan barcha qisqarmaydigan kasrlarni o‘sish tartibida chop eting.

Dastur kodi:	Dastur natijasi:

#include <iostream>
using namespace std;
int gcd(int a, int b) {
if (b == 0) {
return a;
}
return gcd(b, a % b);
}void print(int N) {
for (int d = 1; d <= N; d++) {
for (int n = 0; n <= d; n++) {
if (gcd(n, d) == 1) {
cout << n << "/" << d << endl;
}
}
}
}int main() {
int N;
cout << " sonni kiriting N: ";
cin >> N;
print(N);
return 0; }
	Sonni kiriting:
	7
	0/1
	1/1
	1/2
	1/3
	2/3
	1/4
	3/4
	1/5
	2/5
	3/5
	4/5
	1/6
	5/6
	1/7
	2/7
	3/7
	4/7
	5/7
	6/7

12.2-masala. 2 ta sonning o‘rta arifmetigi va geometrigini hisoblovchi MEAN nomli funksiya hosil qiling. MEAN funksiyasi orqali A, B, C, D sonlaridan (A,B), (A,C), (A,D) juftliklarining o‘rta arifmetigi va geometrigini hisoblovchi dastur tuzilsin.

Dastur kodi:	Dastur natijasi:

#include<iostream>
#include<math.h>
using namespace std;
void MEAN(float a, float b, float *arif, float *geom)
{
*arif = (a + b) / 2;
*geom = sqrt(a * b);
}int main(){
	a=45 b=12 c=36 d=45
	a va b o‘rta arifmetigi=28.5 geometrigi=23.2379
	a va c o‘rta arifmetigi=40.5 geometrigi=40.2492

float a, b, c, d, arif, geom;
cout << "a="; cin >> a;
cout << "b="; cin >> b;
cout << "c="; cin >> c;
cout << "d="; cin >> d;
MEAN(a, b, &arif, &geom);
cout << "a va b o‘rta arifmetigi=" << arif << " geometrigi=" << geom << endl;
MEAN(a, c, &arif, &geom);
cout << "a va c o‘rta arifmetigi=" << arif << " geometrigi=" << geom << endl;
MEAN(a, d, &arif, &geom);
cout << "a va d o‘rta arifmetigi=" << arif << " geometrigi=" << geom << endl;
}
	a va d o‘rta arifmetigi=45
	geometrigi=45

12.3-masala. O‘ngga siklik siljishni amalga oshiruvchi ShiftRight(A,B,C) funksiyasini hosil qiling. Ya’ni A ning qiymati B ga, B nin C ga, C ning qiymati A ga o‘tib qolsin. Bu funksiya orqali (A1, B1, C1) va (A2,B2,C2) sonlarini siljiting.

Dastur kodi:	Dastur natijasi:

#include<iostream>
using namespace std;
void ongga_siljit(int *a, int *b, int *c)
{
int k;
k = *c;
*c = *b;
*b = *a;
*a = k;
}int main(){
int a, b, c;
cout << "a="; cin >> a;
cout << "b="; cin >> b;
cout << "c="; cin >> c;
cout << "Siltilgan holat \n";
	a=1 b=2 c=3
	siljitilgan holat
	a=3 b=1 c=2

```
    ongga_siljit(&a, &b, &c);
    cout << "a=" << a << endl;
    cout << "b=" << b << endl;
    cout << "c=" << c << endl;
}
```

12.4-masala. MonthDays funksiyasidan foydalangan xolda, PrevDate(D,M,Y) funksiya dasturini hosil qiling. Funksiya berilgan sanadan oldingi sanani aniqlasin, D-kun, Y-yil, M-oyini qaytarsin.

Dastur kodi:

```
#include<iostream>
using namespace std;
int kabisayilimi( int yil )
{
 if( (yil % 4 == 0 && yil % 100 != 0) or
yil % 400 == 0)
 {
  return 1;
 }
 else return 0;
}int oykunlari(int yil, int oy)
{
 int kun;
 switch( oy )
 {
 case 1:
 case 3:
 case 5:
 case 7:
 case 8:
 case 10:
 case 12:kun = 31; break;
 case 2: kun = 28 + kabisayilimi( yil );
break;
 default: kun = 30;
 }
 return kun;
```

Dastur natijasi:

```
Yil=2023
Oy=9
Kun=10
Bu kundan oldingi kun:
9 / 9/ 2023
```

```
}void avvalgikun(int *yil, int *oy, int
*kun)
{
  *kun = *kun -1;
  if( *kun == 0)
  {
   *oy = *oy — 1;
   *kun = oykunlari( *yil, *oy );
   if( *oy == 0)
   {
    *oy = 12;
    *yil = *yil -1;
    *kun = 31;
   }
  }
}int main()
{
 int yil, oy, kun;
 cout << "yil="; cin >> yil;
 cout << "oy="; cin >> oy;
 cout << "kun="; cin >> kun;
 avvalgikun( &yil, &oy, &kun);
 cout << kun << " / " << oy << " / " << yil;
}
```

12.5-masala.S1,S2 va S3 satrlari berilgan. S1 satridagi barcha S2 satrini S3 satriga o‘zgartiruvchi programma tuzilsin.

Dastur kodi:	Dastur natijasi:

`#include<iostream>`
`#include<string>`
`using namespace std;`
`int find(string s1, string s2)`
`{`
`int index=-1;`
`if(s1.size()<s2.size()) return -1;`
`for(int i=0; i<=s1.size()-s2.size(); i++)`
`{`
`bool tengmi=true;`
	Satrlarni kiriting:
	S1=o‘zbekiston vatanim mening
	S2=vatanim
	S3=dunyom
	O‘zbekiston dunyom mening

```
            for(int j=0; j<s2.size(); j++)
            {
                if(s1[i+j]!=s2[j])
                {
                    tengmi=false;
                    break;
                }
            }
            if(tengmi)
            {
                index=i;
                break;
            }
        }
        return index;
    }int main(){
        string s1,s2,s3;
        cout<<"s1="; getline(cin,s1);
        cout<<"s2="; getline(cin,s2);
        cout<<"s3="; getline(cin,s3);
        string ans;
        while(find(s1,s2)!=-1)
        {
            ans.append(s1,0,find(s1,s2));
            ans+=s3;

        ans.append(s1,find(s1,s2)+s2.size(),
s1.size());
            s1=ans;
            ans=" ";
        }
        cout<<s1;
    }
```

12.6-masala. Karantindan so‘ng bir maktabda yangi uchta matematikaga yo‘naltirilgan sinf ochish va ular uchun yangi partalar sotib olishga qaror qilindi. Har bir partada ikki kishi o‘tirishi mumkin.

Uchta sinfda ham o‘quvchilar soni aniq. Hammaga parta yetishi uchun eng kamida nechta parta sotib olish kerakligini aniqlaydigan dastur tuzing.

Dastur kodi:	Dastur natijasi:
#include<iostream> using namespace std; int soni(int n) {if(n%2==0) return n/2; else return (n+1)/2;}int main(){ int a,b,c; cin>>a>>b>>c; cout<<soni(a)+soni(b)+soni(c);}	Sonlarni kiriting: 45 12 62 60 ta parta kerak

14-mavzu. Rekursiv va qayta yuklanuvchi funksiyalar.

Odatda, funksiyani o‘zlashtirishda kompilyator funksiya operatorlarini saqlash uchun xotirada faqat bitta katakcha blokini saqlaydi. Funksiya chaqirilgandan so‘ng, dasturni boshqarish ushbu operatorlarga o‘tkaziladi va funksiyadan qaytgach, dastur chaqirilgandan keyingi qatordan dasturning bajarilishi davom etadi. Agar siz ushbu funksiyani 10 marta chaqirsangiz, unda har safar sizning dasturingiz bir xil buyruqlar to‘plamini ishlab chiqadi. Bu shuni anglatadiki, funksiyaning 10 emas, balki faqat bitta nusxasi mavjud.

Ammo funksiya operatorlarini o‘z ichiga olgan xotira maydoniga har bir o‘tish dasturning bajarilishini sekinlashtiradi. Agar funksiya hajmi kichik bo‘lsa (bitta yoki ikkita satrdan iborat bo‘lsa), dasturdan funksiyaga va orqaga o‘tish o‘rniga, kompilyatorga funksiya kodini to‘g‘ridan-to‘g‘ri dasturga kiritish buyrug‘ini bersangiz, samaradorlikda ba’zi yutuqlarni olishingiz mumkin. Samaradorlik deganda, odatda dasturni bajarish tezligi tushuniladi.

Agar funksiya inline kalit so‘zi bilan e’lon qilingan bo‘lsa, kompilyator kompyuter xotirasida funksiyani yaratmaydi, balki uning satrlarini to‘g‘ridan-to‘g‘ri dastur kodiga ko‘chiradi.

Inline funksiyadan foydalanish Listing 12.1 da ko‘rsatilgan.

Listing 12.1. Inline funksiyalaridan foydalanish

```
#include <iostream>
using namespace std;
inline int Double(int);
int main()
int target;
cout << "Enter a number to work with:
cin >> target;
cout << " n";
target = Double(target);
cout << "Target: " << target << endl;
target = Double(target);
cout << "Target: " << target << endl;
target = Double(target);
cout << "Target: " << target << endl;
return 0;
```

```
}
int Double(int target)
{
 return 2•target;
}
```

Natija:

```
Enter a number to work with: 20
Target: 40
Target: 80
Target: 160
```

Tahlil:

3-qatorda int tipidagi parametrni qabul qiladigan va int tipidagi qiymatni qaytaradigan o‘rnini bosuvchi double () funksiyasi e’lon qilinadi. Ushbu e’lon boshqa har qanday prototipga o‘xshaydi, faqat qaytish tipidan oldin inline kalit so‘zi mavjud.

Ushbu prototipni kompilyatsiya qilish natijasi dasturdagi satrlarni almashtirishga teng:

```
target = 2 * target;
```

double()funksiyasini chaqirish:

```
target = Double(target);
```

Inline kalit so‘zi kompilyator uchun funksiya kodini chaqiruv joyidagi dasturga nusxalash uchun xizmat qiladi.

Funksiya o‘zini o‘zi chaqirishi to‘g‘ridan-to‘g‘ri yoki bilvosita bo‘lishi mumkin. Bu jarayonga **rekursiya** deb ataladi. Agar funksiya o‘zini o‘zi chaqirsa **bevosita rekursiya** deb ataladi. Agar funksiya boshqa funksiyani chaqirsa, u holda birinchisini keltirib chiqaradigan bo‘lsa, unda bu holda **bilvosita rekursiya** sodir bo‘ladi.

Ba’zi muammolar rekursiya yordamida osongina hal qilinadi. Shunday qilib, rekursiya ma’lumotlar bo‘yicha ma’lum bir funksiya bajarilgan va keyin olingan natijalar bo‘yicha xuddi shu funksiya bajarilgan hollarda foydalidir.

Rekursiyaning ikkala turi (to‘g‘ridan-to‘g‘ri va bilvosita) ikkita rolda paydo bo‘ladi: ba’zilari oxir-oqibat tugaydi va qaytib keladi, boshqalari esa hech qachon tugamaydi va ish vaqti xatosini keltirib chiqaradi.

Shuni ta’kidlash kerakki, funksiya o‘zini o‘zi chaqirganda, ushbu funksiyaning yangi nusxasi bajariladi. Bunday holda, ikkinchi xil

rekursiyada lokal o‘zgaruvchilar birinchisidagi lokal o‘zgaruvchilardan mustaqil bo‘lib, bir-biriga bevosita ta’sir o‘tkaza olmaydi, hech bo‘lmaganda main() funksiyasidagi lokal o‘zgaruvchilar har qanday boshqa funksiyadagi lokal o‘zgaruvchilarga ta’sir qilishi mumkin.

Misolni ko‘rsatish uchun rekursiya yordamida muammoni hal qilish, Fibonachchi seriyasini ko‘rib chiqing:

1, 1, 2, 3,5, 8, 13, 21, 34…

Qatorning har bir soni undan oldingi ikkita sonning yig‘indisidir. Masalan, Fibonachchi seriyasining 12-sonini aniqlash kerak bo‘lsin.

Ushbu muammoni hal qilishning bir usuli bu qatorni tahlil qilishdir. Birinchi ikkita son 1 ga teng. Har bir keyingi son oldingi ikkitasining yig‘indisiga teng. Shunday qilib, o‘n yettinchi son o‘n oltinchi va o‘n beshinchi son yig‘indisiga teng. Umumiy holda, n — son (n-2)-chi va (n-1)-chi yig‘indisiga teng, agar n>2 bo‘lsa.

Rekursiv funksiyalar uchun rekursiyani tugatish shartini o‘zlashtirish kerak. Dasturda rekursiyani to‘xtatishga majbur qiladigan shart bo‘lishi kerak yoki u hech qachon tugamaydi. Fibonachchi qatorida to‘xtash sharti n < 3 ifodasidir.

Bu quyidagi algoritmdan foydalanadi:

1. Biz foydalanuvchiga Fibonachchi qatoridagi qaysi atamani hisoblash kerakligini ko‘rsatishni taklif qilamiz.

2. Biz fib () funksiyasini chaqiramiz, argument sifatida foydalanuvchi tomonidan belgilangan Fibonachchi qatoridagi sonlarni uzatamiz.

3. fib () funksiyasida argument tahlili (n) amalga oshiriladi. Agar n < 3 bo‘lsa, funksiya 1 qiymatini qaytaradi; aks holda fib () funksiyasi n-2 qiymatini argument sifatida o‘tkazib, o‘zini o‘zi chaqiradi (rekursiv), n-1 qiymatini argument sifatida uzatadi va keyin yig‘indini qaytaradi.

Agar siz fib (1) funksiyasini chaqirsangiz, u 1 ni qaytaradi. Agar siz fib (2) funksiyasini chaqirsangiz, u ham 1 ni qaytaradi. Agar siz fib (3) funksiyasini chaqirsangiz, u fib (2) va fib (1) funksiyalari tomonidan qaytarilgan qiymatlar yig‘indisini qaytaradi. fib (2) funksiyasini chaqirish 1 qiymatini va fib (1) funksiyasini chaqirish 1 qiymatini qaytarganligi sababli, fib (3) funksiyasi 2 qiymatini qaytaradi.

Agar siz fib (4) funksiyasini chaqirsangiz, u fib (3) va fib (2) funksiyalari tomonidan qaytarilgan qiymatlar yig‘indisini qaytaradi. Biz allaqachon fib (3) funksiyasi 2 qiymatini qaytarishini va fib (2)

funksiyasi 1 qiymatini qaytarishini aniqladik, shuning uchun fib (4) funksiyasi bu sonlarni qo‘shadi va Fibonachchi qatorining to‘rtinchi elementi 3 qiymatini qaytaradi.

Keling, keyingi qadamni ko‘ramiz. Agar siz fib(5) funksiyasini chaqirsangiz, u fib (4) va fib (3) funksiyalari tomonidan qaytarilgan qiymatlar yig‘indisini qaytaradi. fib(4) funksiyasi 3 qiymatini va fib (3) funksiyasi 2 qiymatini qaytaradi, shuning uchun yig‘indi 5 soniga teng bo‘ladi.

Ushbu usul bu muammoni hal qilishning eng samarali usuli emas. Masalan, fib(20) funksiyasini chaqirganda, fib() funksiyasi 13529 marta chaqiriladi. Agar siz Fibonachchi qatorining ko‘p elementlarini chaqirsangiz, sizda yetarli xotira bo‘lmasligi mumkin. Har bir fib() funksiyasi chaqirilganda, ma’lum bir xotira maydoni saqlanadi. Funksiyadan qaytgach, xotira bo‘shatiladi. Ammo rekursiv murojaatlar bilan xotiraning barcha yangi sohalari zaxiralanadi va ushbu yondashuv bilan tizim xotirasi juda tez tugashi mumkin. fib() funksiyasini amalga oshirish Listing 12.2 da ko‘rsatilgan.

Listing 12.2 da keltirilgan dasturni ishga tushirganda, Fibonachchi qatorining kichik elementlarini (15 dan kam) o‘rnating. Ushbu dastur rekursiyadan foydalanganligi sababli katta xotira ishlatishi mumkin.

Listing 12.2. **Fibonachchi sonini topish uchun rekursiyadan foydalanishga misol**

```
#include <iostream>
using namespace std;
int fib (int n);
int main()
{
int n, answer;
cout << "Enter number to find: "; cin >> n;
cout << "\n\n";
answer = fib(n);
cout << answer << " is the " << n << "th Fibonacci number\n";
return 0;
}
int fib (int n)
{
cout << "Processing fib(" << n << ")… ";
```

```
if (n < 3 )
{
cout << "Return 1!\n";
return (1);
}
else
{
cout << "Call fib(" << n-2 << ") and fib(" << n-1 << ").\n";
return( fib(n-2) + fib(n-1));
}
}
```

Natija:

```
Enter number to find: 6
Processing fib(6)… Call fib(4) and fib(5).
Processing fib(4)… Call fib(2) and fib(3).
Processing fib(2)… Return 1!
Processing fib(3)… Call fib(1) and fib(2).
Processing fib(1)… Return 1!
Processing fib(2)… Return 1!
Processing fib(5)… Call fib(3) and fib(4).
Processing fib(3)… Call fib(1) and fib(2).
Processing fib(1)… Return 1!
Processing fib(2)… Return 1!
Processing fib(4)… Call fib(2) and fib(3).
Processing fib(2)… Return 1!
Processing fib(3)… Call fib(1) and fib(2).
Processing fib(1)… Return 1!
Processing fib(2)… Return 1!
8 is the 6th Fibonacci number
```

Tahlil:

Ba’zi kompilyatorlar cout obyekti bilan ifodalardagi operatorlardan foydalanishda qiyinchiliklarga duch kelishadi. Agar siz 23-qatorda ogohlantirish olsangiz, 23-qator uchun ayirish amalini qavs ichiga yozing:

```
cout << "Call fib(" << (n-2) << ") and fib(" << n-1 << ").\n";
```

7-qatorda dastur kerakli qator elementining sonini kiritishni taklif qiladi va uni n o‘zgaruvchiga o‘zlashtiradi, keyin fib() funksiyasi n

argumenti bilan chaqiriladi. Dasturning bajarilishi fib() funksiyasiga o‘tadi, bu yerda 15-qatorda ushbu argument ekranda ko‘rsatiladi.

16-qatorda argument 3 sonidan kichik emasligi tekshiriladi va agar shunday bo‘lsa, fib() funksiyasi 1 qiymatini qaytaradi. Aks holda, n-2 va n-1 argumentlari bilan fib() funksiyasini chaqirganda qaytarilgan qiymatlar yig‘indisi olinadi. Shunday qilib, ushbu dastur fib() funksiyasining takrorlashi sifatida ifodalanishi mumkin. Qiymatlarni darhol qaytaradigan yagona murojaatlar fib (1) va fib(2) funksiyalariga murojaat qilishdir.

Ushbu misolda n o‘zgaruvchisi 6 qiymatiga teng, shuning uchun fib (6) funksiyasi main () funksiyasidan chaqiriladi. Dasturning bajarilishi fib() funksiyasining tanasiga o‘tadi va 16-qatorda uzatilgan argumentning qiymati 3 soni bilan taqqoslanadi. 6 soni 3 sonidan katta bo‘lgani uchun, fib(6) funksiyasi fib(4) va fib(5) funksiyalari tomonidan qaytarilgan qiymatlar yig‘indisini qaytaradi:

```
return( fib(n-2) + fib(n-1));
```

Bu shuni anglatadiki, fib(4) va fib(5) funksiyalariga kirish amalga oshiriladi (chunki n o‘zgaruvchisi 6 soniga teng, fib(n-2)=fib(4), fib(n — 1) esa fib(5) ga teng). Shundan so‘ng, hozirda dastur boshqaruvi berilgan fib(6) funksiyasi murojaatlar qanday qaytarilishini kutadi. Qiymatlarning qaytishini kutgandan so‘ng, bu funksiya ushbu ikki qiymatning yig‘indisi natijasini qaytaradi.

15-mavzu. C++ da fayl tushunchasi. Fayldan o‘qish yozish funksiyalari. Fayllar ustida amallar.

Ma’lumotlarni saqlab qo‘yish uchun, tashqi xotiraning nomlangan qismiga fayl deyiladi. Bunday fayllar fizik fayllar deyiladi.

Mantiqiy fayllar. Fizik fayllar bilan ishlash uchun, programmalashtirish tillarida maxsus strukturalashgan, toifalangan fayllar kiritilgan. Bunday fayllar mantiqiy (logicheskiy) fayllar deyiladi. Mantiqiy fayllar, hech qanday fizik xotirani band qilmasdan ma’lumotlarning mantiqiy modelini o‘zida saqlaydi.

Fizik va mantiqiy fayllar bir — biri bilan fopen funksiyasi orqali bog‘lanadi.

Fayl bir nechta elementdan tashkil topgan bo‘lganligi uchun, faqat fayl ko‘rsatkichi ko‘rsatayotgan elementga murojaat qilish mumkin. Fayldan o‘qish yoki yozish mumkin bo‘lgan o‘rinni ko‘rsatuvhi elementga fayl ko‘rsatkichi deyiladi. Fayldan ma’lumot o‘qiganda yoki yozganda fayl ko‘rsatkichi avtomat ravishda o‘qilgan yoki yozilgan bayt miqdoricha siljiydi. Fayl ko‘rsatkichini magnitafon galovkasiga o‘xshatish mumkin.

Binar fayl — har xil ob’yektlarni ifodalovchi baytlar ketma — ketligidir. Ob’yektlar faylda qanday ketma — ketlikda joylashganini programmaning o‘zi aniqlashi lozim. Fayllar bilan ishlovchi funksiyalardan foydalanish uchun <stdio.h> sarlavha faylini dasturga qo‘shish kerak bo‘ladi.

Fayldan ma’lumotlarni o‘qish yoki yozish uchun ochish fopen funksiyasi orqali amalga oshiriladi.

FILE * fopen (const char * filename, const char * mode);

filename — o‘zgaruvchisi char toifasidagi satr bo‘lib, faylning to‘liq nomini ko‘rsatishi lozim

(filename = "D:\c++\misol.txt"). Agar faylning faqat nomi ko‘rsatilgan bo‘lsa, fayl joriy katalogdan qidiriladi (filename = "misol.txt").

mode — o‘zgaruvchisi ham char toifasidagi satr bo‘lib, faylni qaysi xolatda ochish lozimligini bildiradi. mode qiymati faylning ochilish xolati faylni yozish uchun ochish. falename o‘zgaruvchisida ko‘rsatilgan fayl hosil qilinadi va unga ma’lumot yozish mumkin"w" bo‘ladi. Agar fayl oldindan bor bo‘lsa (ya’ni oldin hosil qilingan bo‘lsa), faylning ma’lumotlari o‘chiriladi va yangi bo‘sh fayl faqat yozish uchun ochiq holda bo‘ladi. Fayl o‘qish uchun ochiladi. Agar fayl

oldindan mavjud bo‘lmasa,"r" xatolik sodir bo‘ladi. Ya’ni ochilishi lozim bo‘lgan fayl oldindan hosil qilingan bo‘lishi shart.

Faylga yangi ma’lumotlar qo‘shish — kiritish uchun ochiladi."a"Yangi kiritilgan ma’lumotlar fayl oxiriga qo‘shiladi. Agar fayl
oldindan mavjud bo‘lmasa, yangi fayl hosil qilinadi. Yozish va o‘qish uchun faylni ochish. Agar fayl oldindan bor bo‘lsa (ya’ni oldin hosil qilingan bo‘lsa), faylning ma’lumotlari "w+"o‘chiriladi va yangi bo‘sh fayl yozish va o‘qish uchun ochiqholda bo‘ladi. "r+"Oldindan mavjud bo‘lgan faylni o‘qish va yozish uchun ochish. Fayl ma’lumotlarni o‘qish va yangi ma’lumot qo‘shish uchun "a+" ochiladi. fseek, rewind faylni ochishda xatolik sodir bo‘lsa, fopen funksiyasi NULL qiymat qaytaradi.

Ochilgan faylni yopish uchun fclose funksiyasi ishlatiladi.

int fclose (FILE * stream);

Faylni yopishda xato sodir bo‘lmasa, fclose funksiyasi nol qiymat qaytaradi. Xato sodir bo‘lsa, EOF — fayl oxiri qaytariladi.

Faylga ma’lumot yozish va o‘qish size_t fread (void * ptr, size_t size, size_t n, FILE * stream); fread funksiyasi, fayldan ptr ko‘rsatkichi adresiga size xajmdagi ma’lumotdan n tani o‘qishni amalga oshiradi. Agar o‘qish muvoffaqiyatli amalga oshsa fread funksiyasi o‘qilgan bloklar soni n ni qaytaradi. Aksholda nol qaytariladi size_t fwrite (const void * ptr, size_t size, size_t n, FILE * stream); fwrite funksiyasi, faylga ptr ko‘rsatkichi adresidan boshlab size xajmdagi ma’lumotdan n tani yozishni amalga oshiradi.

15.1. Listing Fayl bilan ishlashga misol.

```
#include <iostream.h>
#include <stdio.h>
int main()
{
int n = 5;
double d = 10.77;
char s[20] = "buxdu.uz";
FILE *f;
// binar faylni yozish uchun ochamiz
f = fopen("misol.txt", "wb");
fwrite(&n, sizeof(int), 1, f); // n sonini faylga yozish
fwrite(&d, sizeof(double), 1, f); // d sonini faylga yozish
```

```
// satrni faylga yozish
fwrite(s, sizeof(char), strlen(s) + 1, f);
fclose(f); // faylni yopish
n = 0; d = 0;
// binar faylni o‘qish uchun ochamiz
f = fopen("misol.txt", "rb");
fread(&n, sizeof(int), 1, f); // n sonini fayldan o‘qish
fread(&d, sizeof(double), 1, f); // d sonini fayldan o‘qish
// satrni fayldan o‘qish
fread(s, sizeof(char), strlen(s) + 1, f);
fclose(f); // faylni yopish
cout << n << endl;
cout << d << endl;
cout << s << endl;
system (“pause”);
return 0;
```

Yuqoridagi misolda satrni yozish va o‘qish uchun quyidagicha kod ishlatildi:

fwrite(s, sizeof(char), strlen(s) + 1, f);
fread (s, sizeof(char), strlen(s) + 1, f);

Buning kamchiligi s satridagi har bir belgi alohida — alohida faylga yozildi va o‘qildi.

Bu masalani quyidagicha hal qilish mumkin edi:

fwrite(s, sizeof(s), 1, f);
fread (s, sizeof(s), 1, f);

Ammo bu usulning ham kamchiligi bor. Ya’ni s satri belgilar i soni massiv o‘lchamidan kam bo‘lgan holda, keraksiz ma’lumotlarni saqlash va o‘qish sodir bo‘ladi. Fayldan ma’lumot o‘qiganda yoki yozganda fayl ko‘rsatkichi avtomat ravishda o‘qilgan yoki yozilgan bayt miqdoricha siljiydi. Fayl ko‘rsatkichining kelgan joyini aniqlash uchun ftell funksiyasi ishlariladi.

long int ftell (FILE * stream);

Fayl ko‘rsatkichini siljit ish uchun fseek funksiyasi ishlatiladi.

int fseek (FILE * stream, long int offset, int whence);

Bu funksiya offset da ko‘ratilgan bayt miqdoricha siljishni amalga oshiradi. whence o‘zgaruvchisi quyidagi qiymatlarni qabul qilishi mumkin:

O‘zgarmas whence Izoh

SEEK_SET 0 Fayl boshiga nisbatan siljitish fayl ko‘rsatkichining joriy SEEK_CUR 1 xolatiga nisbatan siljitish SEEK_END 2 Fayl oxiriga nisbatan siljit ish Agar whence = 1 bo‘lsa (SEEK_CUR), offset musbat (o‘ngga siljish) yoki manfiy (chapga siljish) bo‘lishi mumkin. Fayl ko‘rsatkichini faylning boshiga o‘rnatish uchun rewind funksiyasi ishlatiladi.

void rewind (FILE * stream);

Bu amalni fayl ko‘rsatkichini siljit ish orqali ham amalga oshirish mumkin.

fseek (f, 0, SEEK_SET);

Agar faylda faqat butun sonlar yozilgan bo‘lsa, uning k — elementiga murojaat quyidagicha bo‘ladi.

fseek (f, sizeof(int) * (k — 1), SEEK_SET);

fread (&n, sizeof(int), 1, f);

Fayl oxirini aniqlash uchun feof funksiyasi ishlatiladi.

int feof (FILE * stream);

feof funksiyasi fayl ko‘rsatkichi fayl oxirida bo‘lsa, noldan farqli qiymat qaytaradi.

Boshqa hollarda nol qaytaradi.

15.2-Listing. n natural soni berilgan. Elementlari n ta butun sondan iborat bo‘lgan faylni hosil qiluvchi va ekranga chiqaruvchi programma tuzilsin.

```
#include <iostream.h>
#include <stdio.h>
int main()
{
int n, k;
FILE *f;
f = fopen("binar", "wb+");
// binar faylni yozish va o‘qish uchun ochish
if (f == NULL)
{
cout << "Faylni hosil qilishda xato bo‘ldi";
return 1;
}
cout << "n="; cin >> n;
for (int i = 0; i < n; i++)
{
```

```
cin >> k;
fwrite(&k, sizeof(k), 1, f);
}
// fayl ko‘rsatkichini fayl boshiga qo‘yish
rewind(f);
while (fread(&k, sizeof(k), 1, f))
{
//fayl boshidan fayl ko‘rsatkichi turgan o‘ringacha bo‘lgan baytlar
int bayt = ftell(f);
cout << k <<" ftell(f)=" << bayt << endl;
}
fclose(f);
system (“pause”);
return 0;
}
```

15.3-Listing. Misol. n natural soni berilgan. Elementlari n ta butun sondan iborat bo‘lgan faylni hosil qiluvchi va juft elementlar ini 2 marta orttiruvchi dasturi tuzilsin.

```
#include <iostream.h>
#include <stdio.h>
int main()
{
int n, k;
FILE *f;
// binar faylni yozish va o‘qish uchun ochish
f = fopen("binar", "wb+");
if (f == NULL)
{
cout << "Faylni hosil qilishda xato bo‘ldi";
return 1;
}
cout << "n="; cin >> n;
for (int i = 0; i < n; i++)
{
cin >> k;
fwrite(&k, sizeof(k), 1, f);
}
// fayl ko‘rsatkichini fayl boshiga qo‘yish
```

```
rewind(f);
while (!feof(f)) // fayl oxiri uchramasa bajar
{
fread(&k, sizeof(k), 1, f);
if (k % 2 == 0 )
{
k *= 2;
// fayl ko‘rsatkichini sizeof(int) bayt chapga surish
fseek(f, -sizeof(int), SEEK_CUR);
fwrite(&k, sizeof(int), 1, f);
// fayl ko‘rsatkichini o‘rnatish
fseek(f, ftell(f), SEEK_SET);
}
}
cout << "fayl elementlari\n";
rewind(f);
while (fread(&k, sizeof(k), 1, f))
cout << k << endl;
fclose(f);
return 0;
}
```

Yuqoridagi misolni quyidagicha yechish ham mumkin.

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
int n, k;
FILE *f;
f = fopen("binar", "wb+");
cout << "n="; cin >> n;
for (int i = 0; i < n; i++)
{
cin >> k;
fwrite(&k, sizeof(k), 1, f);
}
// fayl ko‘rsatkichini fayl boshiga qo‘yish
rewind(f);
```

```
while (!feof(f))
{
// fayl ko‘rsatkichi o‘rnini eslab qolish
int pos = ftell(f);
fread(&k, sizeof(k), 1, f);
if (k % 2 == 0 )
{
k *= 2;
// fayl ko‘rsatkichini oldingi xolatiga o‘rnatish
fseek(f, pos, SEEK_SET);
fwrite(&k, sizeof(int), 1, f);
// fayl ko‘rsatkichi o‘rnini sizeof(int) ga surish
pos += sizeof(int);
fseek(f, pos, SEEK_SET);
}
}
cout << "fayl elementlari\n";
rewind(f);
// fayl elementlarini chiqarish
while (fread(&k, sizeof(k), 1, f))
cout << k << endl;
fclose(f);
system (“pause”);
return 0;
}
```

Mustaqil ishlash uchun masalalar

1. Takrolanmaydigan o‘rin almashtirish A_n^k va takrorlanmaydigan o‘rinlashtirish soni C_n^k mos ravishda $A_n^k = \frac{n!}{(n-k)!}$ va $C_n^k = \frac{n!}{k!(n-k)!}$ formulalar bilan topiladi. n va k berilgan holda ushbu kattaliklarni topish uchun dastur yozing.

2. Natural sonning kub ildizining butun qismini va qoldig‘ini topish dasturini yozing (kub ildizining butun qismini olish). Ikkidan katta bo‘lgan ixtiyoriy natural n-ildiz uchun masalani bajaring.

3. Arifmetik kvadrat. n x n kvadrat matritsani shunday to‘ldiringki, birinchi ustun va birinchi qatorning barcha sonlari 1 ga teng, qolgan sonlarning har biri yuqori va chap qo‘shnilarning yig‘indisiga teng. Ushbu o‘lchamdagi matritsani ekranga chiqaring.

4. Uchta son berilgan. Bunday uzunlikdagi kesmalardan uchburchak hosil qilish mumkinligini aniqlang. Agar mavjud bo‘lsa, uchburchakning turini (to‘g‘ri burchakli, o‘tmas burchakli, o‘tkir burchakli) aniqlang.

5. Pifagor sonlari. a, b va c uchta natural sonlar Pifagor uchligini hosil qiladi, agar $c^2=a^2+b^2$ bo‘lsa. Asosiy Pifagor uchligi deyiladi, agar uning sonlarining eng katta umumiy bo‘luvchisi birga teng bo‘lsa. Masalan, 3, 4, 5 — asosiy uchlik, 6, 8, 10-olingan uchlik. Berilgan max sonidan oshmaydigan barcha asosiy Pifagor uchliklarini toping.

6. Tomonlarining uzunligi butun sonlar bo‘lgan to‘rtburchak berilgan. Berilgan to‘rtburchakni har safar kesilganda eng katta maydonli kvadrat tomoni, joriy to‘rtburchakning umumiy tomoni kesilgan bo‘lsa kesish mumkin bo‘lgan kvadratlar sonini (har bir kvadrat tomonlarining uzunligi butun sonlar) toping, agar.

7. Berilgan butun sonli to‘rtburchak maydonining bo‘lishi mumkin bo‘lgan butun sonli yon tomonlarining uzunliklarini toping. Masalan, 12 ga teng maydon uchun 1 x 12, 2 x 6, 3 x 4 uchta to‘rtburchaklar mos keladi. Hajmi V bo‘lgan va qirralari natural sonlar bilan ifodalangan, asosi har xil to‘rtburchakli parallelepipedlarni topish uchun dastur yozing. Agar siz qirralarni almashtirsangiz, bir-biridan olingan parallelepipedlar bir xil deb hisoblanadi.

8. L. Kerroll o‘z kundaligida hech bo‘lmaganda uchta teng maydonli to‘g‘ri burchakli uchburchaklarni topishga harakat qilganini

yozgan, ularning tomonlari uzunligi natural sonlar bilan ifodalangan bo‘lar edi. Agar bunday uchburchaklar mavjudligi ma’lum bo‘lsa, ushbu muammoni hal qilish uchun dastur tuzing. Maydoni berilgan s sonidan oshmaydigan barcha to‘g‘ri burchakli uchburchaklarni (yon uzunliklar natural sonlar bilan ifodalanadi) topadigan dastur yozing.

9. Ikkita natural sonning kublari yig‘indisi sifatida ifodalanishi mumkin bo‘lgan eng kichik natural sonni bir necha usulda toping.

10. Mushuk o‘n uchta sichqon bilan o‘ralganligini tush ko‘radi. Ulardan o‘n ikkitasi kulrang, biri oq. Agar u har o‘n uchdan birini iste’mol qilsa va oq sichqonchani oxirgi marta iste’mol qilishi kerak bo‘lsa, u qaysi sichqonchadan hisoblashni boshlaydi? Vazifani hisoblash uchun n sichqonlarning umumiy soni va hisoblash m sichqonlargacha amalga oshiriladi deb hisoblang.

11. Ikki natural sonning eng katta umumiy bo‘luvchisini toping. Ikkita variantni amalga oshiring: rekursiv va rekursiyasiz. n natural sonlar uchun vazifani bajaring.

12. Berilgan ikkita natural sonning o‘zaro toq ekanligini aniqlang. Uchta son o‘zaro toq yoki yo‘qligini tekshiradigan funksiyani tuzing.

13. Qiymati 1 dan n gacha bo‘lgan barcha natural sonlarning eng kichik umumiy karralisini topadigan funksiyani tuzing (n-funksiya parametri).

14. Berilgan natural sonning barcha bo‘linuvchilarini, shuningdek ularning soni va yig‘indisini toping.

15. Eratosfen panjarasi (Eratosfen g‘alviri) yordamida birinchi n ta tasodifiy sonlar to‘plamini yaratadigan dastur yozing.

16. 2 ga farq qiladigan ikkita toq tub sonlar egizaklar deb ataladi. Masalan, 5 va 7 raqamlari. [2; 1000] segmentdagi barcha egizak raqamlarni topadigan dasturni yozing.

17. Bir kuni matematik S.Ulam qog‘oz varag‘ini kataklarga ajratdi va markazga 1 ni yozib, ketma-ket barcha natural sonlarni spiral shaklida soat strelkasiga qarama-qarshi yo‘nalishda yozishni boshladi. Tez orada tub sonlar mantiqiy tartibda tizilib, qiziqarli naqsh hosil qildi. Keyinchalik bu naqsh tadqiqot obyektiga aylandi va Ulam dasturxoni deb nomlandi. 100 x 100 katakli ulcham dasturxonini namoyish etadigan dastur tuzing (tub sonlar o‘rniga "*" yulduzchani chiqaring).

18. Mukammal son -bu o‘zidan kichik bo‘lgan bo‘luvchilarning yig‘indisiga teng bo‘lgan son. Masalan, 28=1+2+4+7+14. Berilgan natural sonning mukammalligini aniqlang.

19. Ikkita natural sonlar do‘stona sonlar deb ataladi, ularning har biri boshqa sondan kichik bo‘lgan bo‘luvchilarning yig‘indisiga teng bo‘lsa(boshqa sonning o‘zi kirmaydi). Masalan, 220 va 284. Berilgan segmentdagi barcha do‘stona sonlarni toping. Ushbu segmentda eng ko‘p bo‘luvchiga ega sonni topadigan dasturni yozing.

20. Agar sonning kvadratining oxirgi raqamlari sonning kvadratiga teng bo‘lsa natural son avtomorf deyiladi. Masalan, 5, chunki $5^2=25$ yoki 25, chunki $25^2=625$. Ushbu segmentdagi barcha avtomorf sonlarni toping. Sonning kubinchi darajasi uchun shunga o‘xshash vazifani bajaring.

21. Kitobda n sahifalar mavjud. Bunday kitobning barcha sahifalarini betlash uchun zarur bo‘lgan raqamlar sonini toping.

22. Raqamlari kublari yig‘indisiga teng bo‘lgan natural sonlar mavjud. Masalan, 370 raqami, chunki $3^3+7^3+0^3=370$. Raqamlari yig‘indisi kubiga teng bo‘lgan barcha natural sonlarni topadigan dastur tuzing.

23. Turli xil raqamlardan tashkil topgan va to‘liq kvadrat bo‘lgan barcha n-xonali ($2<n<9$) sonlarni toping. Berilgan so‘zda nechta turli harflar borligini hisoblaydigan dastur tuzing.

24. B. Kordemskiy o‘z raqamlarining faktoriallari yig‘indisiga teng bo‘lgan bitta 145 sonini ko‘rsatadi: $145=1!+4!+5!$. Uning yozishicha, ushbu shartni qanoatlantiradigan yana shunday sonlar mavjudmi yoki yo‘qmi noma’lum. Bunday sonlarning barchasini topishga yordam bering.

25. Ikkilik sanoq sistemasida son berilgan. Ushbu sonni o‘nlik sistemasida aniqlang. Ikkilik sanoq sistemasida yozilgan ikkita butun sonni oladigan, ularni qo‘shadigan va natijani ham ikkilik sanoq sistemasida ko‘rsatadigan dastur tuzing.

26. Ikkilik sonni o‘n oltilikka va aksincha aylantirish uchun dastur yozing.

27. Berilgan sonni berilgan aniqlik bilan bo‘lish natijasini toping (ya’ni javobda verguldan keyin berilgan raqamlar soni mavjud).

28. Oddiy kasrni o‘nli kasrga aylantiring. Agar kasr davriy bo‘lsa, unda davr qavs ichida ko‘rsatiladi. Davrni verguldan keyingi 100 raqam ichidan qidiring.

29. Omonat banklariga qo‘yilgan muddatsiz k omonat summasining yillik p foizi to‘lanadi. Agar omonatchi omonatdan pul

olib qo‘ymasa, foizlar har yili tobora ko‘proq miqdorda olinadi. m yildan so‘ng omonat miqdorini toping.

30. [a; b] segmentdan tasodifiy natural sonlar qatorini yaratish formulasini yarating.

31. Sana kun:oy:yil formatida berilgan. n kundan keyingi sanani aniqlang.

32. Sana kun:oy:yil formatida berilgan. Haftaning kunini aniqlang.

33. Tug‘ilgan kun sanasi kun:oy:yil formatida berilgan. Tug‘ilgan haftaning kuni keyingi tug‘ilgan kunning hafta kuniga to‘g‘ri keladigan asr davomidagi barcha yillarni toping.

34. Rombni "*"belgilari yordamida chiqaring. Romb o‘lchami foydalanuvchi tomonidan beriladi.

35. Butun sonlar massivida eng ko‘p uchraydigan elementni toping.

36. Massiv berilgan. Yig‘indisi minimal bo‘lgan ikkita qo‘shni elementni toping.

37. Berilgan massiv elementlari qiymatlarining o‘suvchi yoki kamayuvchi ekanligini tekshiring.

38. Matn berilgan. Ushbu matndagi matn so‘zlarini hamda ularning takrorlashlari sonini chop eting.

39. Berilgan matematik ifodadagi ochiluvchi va yopiluvchi qavslarni to‘g‘ri qo‘yilganligini tekshiruvchi dastur kodini yozing.

40. Elementlari talabaning to‘liq ismi va guruh raqamini o‘z ichiga olgan massiv berilgan. Massivni guruh raqamlari bo‘yicha tartiblang, shunda bitta guruh ichida talabalar alifbo tartibida tartiblanadi.

41. Sonni Arab raqamlaridan Rim raqamlaridagi songa o‘tkazish uchun dastur yozing. Teskari tarjima dasturini yozing.

42. Morze alifbosi yordamida berilgan matnni kodlash uchun dastur yozing.

43. Shaxmat doskasida shohning koordinatalari berilgan. Shoh yurishi mumkin bo‘lgan barcha yo‘llarni aniqlang.

44. Ushbu vazifani ot uchun ham bajaring (43-masala).

45. n ta gugurt berilgan. Foydalanuvchi va kompyuter navbatma-navbat bir nechta gugurtni oladi (har bir harakat uchun 1 dan p gacha). Oxirgi gugurtni olgan kishi yutqazadi. Agar biror qoida mavjud bo‘lsa, kompyuter g‘alaba qozonish strategiyasiga rioya qilishi uchun o‘yin jarayonini amalga oshiring.

46. Berilgan katakdan boshlab ot shaxmat taxtasida o‘tish yo‘llari dasturini tuzing. Ot har bir katakka bir marta yurishi mumkin.

47. Viloyatning tuman raqamlari ro‘yxati berilgan. Har bir tuman uchun uning qo‘shnilari ma’lum. Qo‘shnilar bir guruhda bo‘lmasligi uchun barcha tumanlarni to‘rt guruhga bo‘ling.

48. Matnli fayl kvadrat matritsa prinsipi bo‘yicha yozilgan: bitta satr matritsaning bitta elementidir. Matritsaning o‘lchamini aniqlash va ikki o‘lchovli massivni qurish kerak. Asl matritsani va uning soat yo‘nalishi bo‘yicha 90º aylanish natijasini ko‘rsating.

49. Fayl matn satrini o‘z ichiga oladi. Matndagi har bir harfning takrorlanish chastotasini aniqlang va uni ko‘rsating.

50. Faylda matn satrlari to‘plami mavjud. Har bir so‘zning birinchi harfini bosh harfga o‘zgartiring.

FOYDALANILGAN ADABIYOTLAR

1. O‘zbekiston Respublikasi Prezidentining 2018 yil 13 dekabrdagi “O‘zbekiston Respublikasi davlat boshqaruviga sonli iqtisodiyot, elektron hukumat hamda axborot tizimlarini joriy etish bo‘yicha qo‘shimcha chora-tadbirlar to‘g‘risida”gi PF-5598-son Farmoni.

2. A.A.Xaldjigitov, Sh.F.Madraximov, U.E.Adamboev, Informatika va programmalash. O‘quv qo‘llanma, O‘zMU, 2005 yil, 145 bet.

3. Algorithms, Fourth Edition (Deluxe): Book and 24-Part Lecture Series 1st Edition , Addison-Wesley Professional , USA, 2015.

4. Mirsanov U.M., Toxirov F.J., Norbekov A.O., Djurayeva D.R. Dasturlash. // O‘quv qo‘llanma. Toshkent, 2021. – 152 b.

5. O.Shukurov, F.Qorayev, E.Eshboyev, B.Shovaliyev “Programmalashdan masalalar to‘plami”, Toshkent 2008,160 bet.

6. Ro‘ziyev R.A. Dasturlash asoslari // O‘quv qo‘llanma. — Toshkent, 2020. –159 b.

7. Б. Страуструп, “Язык программирования C++. Специальное издание”,-М.:ООО «Бином-Пресс», 2006.-1104 с.

8. Герберт Шилдт: C++ базовый курс.

9. M.A.Bobojonova “Dasturlash asoslari 1-qism” Buxoro “KAMOLOT” nashriyoti, 2023. - 420 b.

Qaydlar uchun

T.R.Shafiyev, Sh.E.Nazarov, Z.K.Niyozova

DASTURLASH ASOSLARI FANIDAN MASALALAR VA YECHIMLAR (C++)

O‘QUV QO‘LLANMA

Muharrir: *E.Eshov*
Tex.muharrir: *D.Abduraxmonova*
Musahhih: *M.Shodiyeva*
Badiiy rahbar: *M.Sattorov*

Nashriyot litsenziyasi № 022853. 08.03.2022.
Original maketdan bosishga ruxsat etildi: 10.03.2024. Bichimi 60x84. Kegli 16 shponli. “Times New Roman” garnitura 1/16. Ofset bosma usulida. Ofset bosma qog‘ozi.
Bosma tabog‘i 9,25. Adadi 100. Buyurtma № 93

KAMOLOT

“BUXORO DETERMINANTI” MCHJ
bosmaxonasida chop etildi.
Buxoro shahar Namozgoh ko‘chasi 24 uy
Tel:. + 998 91 310 27 22