O’ZBEKISTON RESPUBLIKASI ALOQA, AXBOROTLASHTIRISH VA TELEKOMUNIKATSIYA DAVLAT QO’MITASI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI FARG’ONA FILIALI

”Axborot texnologiyalari” kafedrasi

”C++ da dasturlash”
fanidan

KURS ISHI

Bajardi: 630-12 Guruh talabasi

Hojayev Begzod

Qabul qildi: Bazarbayev M.

Hashimov A

Farg’ona-2015

Mavzu: Kiritilgan N sonini bo’luvchilarini chiqaruvchi dastur

Reja:

1. **Kirish.**
    - **1.1 C++ dasturlash tili tarixi.**
    - **1.2 C++ dasturlash tili tuzilishi.**
2. **Asosiy qism.**
    - **2.1 C++ dasturlash tilida dasturlashning asosiy qismlari.**
    - **2.2Dastur ijro strukturalari.**
    - **2.3 Mantiqiy operatorlar.**
    - **2.4 For takrorlash operatori.**
3. **Dasturiy qism.**
4. **Xulosa.**
5. **Foydalanilgan adabiyotlar.**

1. Kirish.

C++ dasturlash tilining tarixi.

C++ dasturlash tili C tiliga asoslangan. C esa o'z navbatida B va BCPL tillaridan kelib chiqqan. BCPL 1967 yilda Martin Richards tomonidan tuzilgan va operatsion sistemalarni yozish uchun mo'ljallangan edi. Ken Thompson o'zining B tilida BCPL ning ko'p hossalarini kiritgan va B da UNIX operatsion sistemasining birinchi versiyalarini yozgan. BCPL ham, B ham tipsiz til bo'lgan.Yani o'garuvchilarning ma'lum bir tipi bo'lmagan - har bir o'zgaruvchi kompyuter hotirasida faqat bir bayt yer egallagan.O'zgaruvchini qanday sifatda ishlatish esa, yani butun sonmi, kasrli sonmi yoki harfdekmi, dasturchi vazifasi bo'lgan.

C tilini Dennis Ritchie B dan keltirib chihardi va uni 1972 yili ilk bor Bell Laboratoriyasida, DEC PDP-11 kompyuterida qo'lladi. C o'zidan oldingi B va BCPL tillarining juda ko'p muhim tomonlarini o'z ichiga olish bilan bir qatorda o'zgaruvchilarni tiplashtirdi va bir qator boshqa yangiliklarni kiritdi. Boshlanishda C asosan UNIX sistemalarida keng tarqaldi.Hozirda operatsion sistemalarning asosiy qismi C/C++ da yozilmoqda.C mashina arxitekturasiga bog'langan tildir.Lekin yaxshi rejalashtirish orqali dasturlarni turli kompyuter platformalarida ishlaydigan qilsa bo'ladi.

1983 yilda, C tili keng tarqalganligi sababli, uni standartlash harakati boshlandi.Buning uchun Amerika Milliy Standartlar Komiteti (ANSI) qoshida X3J11 tehnik komitet tuzildi.Va 1989 yilda ushbu standart qabul qilindi. Standartni dunyo bo'yicha keng tarqatish maqsadida 1990 yilda ANSI va Dunyo Standartlar Tashkiloti (ISO) hamkorlikda C ning ANSI/ISO 9899:1990 standartini qabul qilishdi. Shu sababli C da yozilgan dasturlar kam miqdordagi o'zgarishlar yoki umuman o'zgarishlarsiz juda ko'p kompyuter platformalarida ishlaydi.

C++ 1980 yillar boshida Bjarne Stroustrup tomonidan C ga asoslangan tarzda tuzildi.C++ juda ko'p qo'shimchalarni o'z ichiga olgan, lekin eng asosiysi u ob'ektlar bilan dasturlashga imkon beradi.

Dasturlarni tez va sifatli yozish hozirgi kunda katta ahamiyat kasb etmoda. Buni ta'minlash uchun ob'ektli dasturlash g'oyasi ilgari surildi.Huddi 70-chi yillar boshida strukturali dasturlash kabi, programmalarni hayotdagi jismlarni modellashtiruvchi ob'ektlar orqali tuzish dasturlash sohasida inqilob qildi.

C++ dan tashhari boshqa ko'p ob'ektli dasturlshga yo'naltirilgan tillar paydo bo'ldi.Shulardan eng ko'zga tashlanadigani Xerox ning Palo Altoda joylashgan ilmiy-qidiruv markazida (PARC) tuzilgan Smalltalk dasturlash tilidir.Smalltalk da hamma narsa ob'ektlarga asoslangan.C++ esa gibrid tildir.Unda C ga o'hshab

strukturali dasturlash yoki yangicha, ob'ektlar bilan dasturlash mumkin.Yangicha deyishimiz ham nisbiydir.Ob'ektli dasturlash falsafasi paydo bo'lganiga ham yigirma yildan oshayapti.

C++ funksiya va ob'ektlarning juda boy kutubhonasiga ega. Yani C++ da dasturlashni o'rganish ikki qismga bo'linadi. Birinchisi bu C++ ni o'zini o'rganish, ikkinchisi esa C++ ning standart kutubhonasidagi tayyor ob'ekt funksiyalarni qo'llashni o'rganishdir.

Dasturlash tili tuzilishi.

1. Alfavit, identifikator, xizmatchi so'zlar.

Alfavit. C++ alfavitiga quyidagi simvollar kiradi.

- Katta va kichik lotin alfaviti xarflari (A,B,..,Z,a,b,…,z)
- Raqamlar: 0,1,2,3,4,5,6,7,8,9
- Maxsus simvollar: " , {} | [] () + - / % \ ; ' . : ?< = > _ ! & * # ~ ^
- Ko'rinmaydigan simvollar ("umumlashgan bushliq simvollari"). Leksemalarni uzaro ajratish uchun ishlatiladigan simvollar (misol uchun bushlik, tabulyatsiya, yangi qatorga o'tish belgilari).

Izohlarda, satrlarda va simvolli konstantalarda boshqa literalar, masalan rus xarflarini ishlatilishi mumkin.

C++ tilida olti khil turdagi leksemalar ishlatiladi: ehrkin tanlanadigan va ishlatiladigan identifikatorlar, khizmatchi suzlar, konstantalar(konstanta satrlar), amallar(amallar belgilari), o'zgaruvchi belgilar.

Identifikator.Identifikatorlar lotin xarflari,ostki chiziq belgisi va sonlar ketma ketligidan iborat buladi. Identifikator lotin xarfidan yoki ostki chizish belgisidan boshlanishi lozim.

Misol uchun:

A1, _MAX, adress_01, RIM, rim

Katta va kichik xarflar farklanadi, shuning uchun ohirgi ikki identifikator bir biridan farq qiladi.

Borland kom'ilyatorlaridan foydalanilganda nomning birinchi 32 xarfi ,ba'zi kom'ilyatorlarda 8 ta xarfi inobatga olinadi. Bu holda NUMBER_OF_TEST va NUMBER_OF_ROOM identifikatorlari bir biridan farq qilmaydi.

Asosiy qism.

C++ dasturlash tilida dasturlashning asosiy qismlari.

C++ sistemasining asosan quyidagi qismlardan iborat. Bular dasturni yozish redaktori, C++ dasturlash tili va standart kutibxonalardir. C++ dasturi ma'lum bir vazifalardan o'tadi. Birinchi dasturni yozish va taxrirlash , ikkinchisi 're'rosessor amallarini bajarish, kom'ilizatsiya, kutubxonalardagi ob'yekt va funksiyalarni dastur bilan bog'lash (link), xotiraga yuklash (load) va bajarish (exusute).

C++ dasturlash tilida ikkita tur sharxlar mavjud. /" bilan boshlanib, "/ bilan tugaydigani bir necha satrlarni egallashi mumkin. Ya'ni bu belgilar orasida qolgan xamma yozuv sharx xisoblanadi.Bu tur sharx CI dan qolgan.C++ dasturlash tilida yangi ko'rinishdagi sharxlar kiritilgan. Bu // bilan boshlanadi va kuchi shu satr oxirigacha saqlanadi. Sharxlar yoki boshqacha qilib aytgandakamentariylar kom'ilyator tomonidan xisobga olinmaydi va xech qanday mashina ijroga qoidaga aylantirilmayda. Sharxlar kerakli joyda, funksiyalardan oldin o'zgaruvchilar e'lonidan keyin yozilganda, dasturni tushinish osonlashadi va keyinchalik funksiya bilan ishlash mantig'ini esga solib turadi.

#include<iostream> bu 're'rosessorga beriladigan buyruqdir. 're'rosessor kom'ilizatsiyadan oldin fayllarni ko'rib chiqadi va kerakli amallarni bajaradi. Unga tegishli bo'lgan buyruqlar # belgisi bilan lekin buyruq oxiriga nuqta vergul (;) qo'yilmaydi. Bu yerda include (kititmoq, qamrab olmoq) buyrig'I iostream faylini asosiy dasturimiz ichiga kiritadi. Bu fayl ichida biz ishlayotgan cout oqim (stream) ob'yektining e'loni berilgan. C++ tilida ekran yoki klavyaturadan kirish / chiqishni bajarmoqchi bo'lgan barcha dasturlar ushbu boshliq (header) faylini yoki uning yangi ko'rinishi include bilan o'z ichiga olishi kerak. Bu kabi fayllarni biz bundan keyin e'lon fayllari deb ataymiz. Chunki bu fayllar ichida funksiya va ob'yektlarning o'zi, ya'ni tanasi berilmay faqatgina e'loni beriladi. Include buyrug'i ikki xil yo'l bilan qo'llanilishi mumkin.

1. **include <iostream>**
2. **include "mening faylim"**

Birinchi usulda e'lon fayli <>qavslar ichida yoziladi.Bunda C++ siistemasi oldindan belgilangan kataloklar ichidan kiradi.Bu usul bilan asosan standart kutibxona fayllari qo'llaniladi.Ikkinchi usulda, fayl nomi qo'shtirnaqlarga olinganda, kiritilishi kerak bo'lgan joriy fayl katalogdan qidiriladi.Bu yo'lbilan dasturchi o'zo yozgan e'lon fayllarini kiritadi. Shuni aytib o'tish kerakki, C++ ning 1998 yilda qabul qilingan standartiga ko'ra, ko'ra ushbu e'lon fayllari yangi ko'rinishga ega ular .h bilan tugamaydi. Bunda, misol uchun bizning iostream.h

faylimiz iostream, CI dan kelgan math.h esa cmath nomiga ega. . Biz bu o`zgarishlarga keyinroq qaytamiz, xozircha esa eski tipdagi e`lon fayllaridan foydalanib turamiz. Int main() xar bir C++ dasturining qismidir. Main dan keyingi () qavslar C++ ning funksiya deb ataluvchi blokining boshlanganligini bildiradi. C++ dasturi bir yoki bir necha funksiyalardan iborat va shulardan aniq bitta funksiya main deb atfuntsiyasi shart. Bunda main dasturi ichida keladigan birinchi funksiya bo`lmasligi ham mumkin. Operatsion sistema dastur ijrosini main() funsiyasidan boshlaydi. Main() dan oldin kelgan int esa main funksiyasidan qaytish qiymati tipini belgilaydi. Bunda int integer, ya`ni butun son deganidir. Main() ning qaytargan qiymati operatsion sistemaga boradi. { qavs funksiya va boshqa bloklar tanasini boshlaydi. Blokni yopish uchun }qavsi ishlatiladi.cout << "Hello World!\n"; satri C++ da ifoda deb ataladi. C++ dagi har bir ifoda ; (nuqta-vergul) bilan tugatilishi shart. Ortiqcha ; bo`sh ifoda deyiladi. Uni qo`yish dastur tezligiga ta`sir qilmaydi. Kirish va chiqish (Input/Output), ya`ni dasturga kerakli ma`lumotlarni kiritish va ular ustida dastur tomonidan bajarilgan amallar natijalarini olish C++ da oqim ob`ektlari ortiqcha funksiya bajarilishi mumkin. Lekin kirish/chiqishni CI dagi kabi funksiyalar bilan ham amalga oshirsa bo`ladi. C++ falsafasiga ko`ra har bir kirish/chiqish jixozi (ekran, printer, klaviatura...) baytlar oqimi bilan ishlagandek qabul qilinadi. Yuqoridagi ifoda bajarilganda bizning "Hello World!" gapimiz standart chiqish oqimi ob`ekti cout ga (cout - console out) jo`natiladi. Normal sharoitda bu oqim ekranga ulangandir. C++ da satrlar (string) qo`shtirnoqlar (") orasida bo`ladi. Bitta harfli literalar esa bitta tirnoq - apostrof (') ichiga olinadi. Misol uchun: 'A'. Bitta harf yoki belgini qo`shtirnoq ichiga olsa u satr kabi qabul qilinadi.<<operatori oqimga kiritish operatori deyiladi. Programma ijro etilganda << operatorining o`ng tomonidagi argument ekranga yuboriladi.Bunda ekranga qo`shtirnoq ("...") ichidagi narsa bosib chiqariladi.Lekin e`tibor bersak, \n belgisi bosilmadi.\ (teskari kasr - backslash) belgisi mahsus ma`noga ega.U o`zidan keyin kelgan belgi oqim buyrug’i yoki manipulyatori ekanligini bildiradi.SHunda \ belgisi bilan undan keyin kelgan belgi buyruq ketma-ketligida aylanadi. Bularning ro`yxatini beraylik:

\n - Yangi satr. Kursor yangi qator boshidan joy oladi.

\t - Gorizontal tabulyatsiya (kursor bir-necha harf o`nga siljiydi).

\v - Vertikal tabulyatsiya (bir-necha satr tashlanib o`tiladi).

\r - qaytish. Kursor ayni satr boshiga qaytadi, ya`ni yangi satrga o`tmaydi.

\a - Komp'yuter dinamiki funksiyalanadi.

\\ - Ekranga teskari kasr belgisini bosish uchun qo`llaniladi.

\" - Экранга qo`shtirnoq belgisini bosish uchun qo`llaniladi.

Return 0; (return - qaytmoq) ifodasi main() funksiyasidan chiqishning asosiy yo`lidir. 0 (nol) qiymatining qaytarilishi operatsion sistemaga ushbu dastur normal bajarilib tugaganini bildiradi.return orqfunksiya qaytadigan qiymat tipi funksiya e`lonidagi qaytish tipi bilan bir xil bo`lishi kerak. Bizda bu e`lon int main(){...} edi va 0 int tipiga mansubdir. Bundan keyin return orqfunksiya qaytarilayotgan ifodani qavs ichiga olamiz. Misol uchun return (6). Bu qavslar majburiy emas, lekin bizlar ularni programmani o`qishda qulaylik uchun kiritamiz.

Endi kichik bir dasturni ko’rib chiqaylik.

```
    //Ushbu dastur ikki butun sonni ko`paytiradi.
# include <iostream.h>
int main()
{
int son A, son B; //o`zgaruvchi e`lonlari
int summa;      //e`lon
cout<< "Birinchi sonni kiriting: ";
cin>> son A; //Birinchi sonni o`qish...
cout<< "Ikkinchi sonni kiriting: ";
cin>> son B; //Ikkinchi sonni o`qish...
summa = son A * son B;
cout<< summa << endl;
cout<< "son A * son B = " << son A * son B << endl;
return (0);
}
```

Экранда:

Birinchi sonni kiriting: 4

Ikkinchi sonni kiriting: 6

Son A * son B = 24

Int son A, son B; ifodasi int tipidagi, ya`ni integar (butun son) bo`lgan ikkita o`zgaruvchini e`lon (declaration) qildik. Agar o`zgaruvchilar tipi bir xilda bo`lsa, yuqoridagi kabi ularni ketma-ket, vergul bilan ayirib yozsak bo`ladi. Keyingi satrda esa int summa; bilan summa nomli o`zgaruvchini e`lon qildik.cout<< "Birinchi sonni kiriting: "; ifodasi bilan ekranga nima qilish kerakligini yozib chiqdik. cin>> son A; anfunktsiya cin kirish oqimi ob`ekti orqfunksiya son A o`zgaruvchisiga klaviaturadan qiymat kiritmoqda. Sonni yozib bo`lgandan so`ng enter ni bosamiz. Normal sharoitda kirish oqimi klaviaturaga bog'langan.Shu tariqa son B ga ham qiymat berdik. Keyin esa summa = son A * son B; bilan biz ikki o`zgaruvchini ko`paytirib, ko`paytma qiymatini summa ga beryapmiz. Bu erdagi "=" va "*" operatorlar ikki argumentli operatorlar deyiladi, chunki ular ikkita operand yoki boshqacha qilib aytganda kirish qiymatlari bilan ishlaydi. Operatorlardan oldin va keyin bo`sh joy qoldirsak, o`qishni osonlashtirgan bo`lamiz. Экranga javobni chiqarganda, cout ga tayyor natijani (summa) yoki matematik ifodaning o`zini berishimiz mumkin. Oxirgi cout ga bir-necha argumentni berdik.Endl (end line - satrni tugatish) bu oqim manipulyatoridir (stream manipulator).Ba`zi bir sistemalar chiqish oqimiga yo`naltirilgan ma`lumotlarning ma`lum bir miqdori yiqilguncha ushbu ma`lumotlarni ekranga bosib chiqarmay, buferda saqlashadi va o`sha chiqish buferi to`lgandan keyingina ma`lumotlarni ekranga yuborishadi.Buning sababi shuki, ekranga bosish nisbatan vaqt jihatdan qimmat amaldir.Agar ma`lumotlar yig'ilib turib, bittada chiqarilsa, dastur ancha tez ishlaydi.Lekin biz yuqoridagi dasturdagi kabi qo`llanuvchi bilan savol-javob qiluvchi programmada yo`l-yo`riqlarimizni berilgan paytning o`zida ekranga bosib chiqarilishini hohlaymiz. Shu sababli biz endl niishlatishimiz kerak.Endl ni biz "\n" buyrug'iga tenglashtirishimiz mumkin. Ya`ni endl ni ishlatganimizda, bufer yoki boshqacha qilib aytganda, xotiradagi ma`lumotni va tinchfunksiyak saqlanish joyidagi informatsiya ekranga bosib chiqarilgandan so`ng, kursor yangi satr boshiga ko`chadi. Agar biz buferni bo`shatmoqchi-yu, lekin kursorni joyida saqlab qolmoqchi bo`lsak, flash manipulyatorini ishlatishimiz lozim. Ifodamizga qaytaylik, cout << "son A * son B = " << son A * son B << endl; ifodasida chiqish ob`ekti bitta, lekin biz unga uchta narsani yubordik. Buni biz oqimga ma`lumotlarni chiqarishni kaskadlash, zanjirlash yoki konkatenatsiya qilish deb ataymiz.Ayni amalni cin (console in) kirish oqimi uchun ham bajara olamiz.xisob-kitoblar chiqish ifodasi ichida ham bajarilishi mumkin, cin << son A *son B << endl; bunga misol. Agar bu yo`lni o`tganimizda, summa o`zgaruvchisi kerakmas bo`lib qolardi. Ushbu dasturda bizda yangi bo`lgan narsalardan biri bu o`zgaruvchi (variable) tushunchasidir. O`zgaruvchilar kompyuter

xotirasidagijoylarga ko`rsatib turishadi. har bir o`zgaruvchi ism, tip, xotirada egallagan joy kattafunksiyagi va qiymatga egadir. O`zgaruvchi ismi katta-kichik xarf, son va past tiredan ( _ - underscore) iboratdir. Lekin sondan boshlana olmaydi. CI/C++ da katta-kichik xarf, ya`ni xarflar registri farqlanadi. Misol uchun A1 va a1 farqli ismlardir.

Amallar jadvali

Arifmetik amallar	Razryadli amallar	Nisbat amallari	Mantiqiy amallar
+ qo’shish	&va	= = teng	&&va
- bo’lish	| yoki	!= teng emas	|| yoki
* ko’paytirish	^ inkor	>katta	! inkor
/ bo’lish	<<chapga surish	>= katta yoki teng	
% modul olish	>>o’ngga surish	<kichik	
- unar minus	~ inkor	<= kichik yoki teng	
+ unar plyus			
++ oshirish			
-- kamaytirish			

Amallar jadvali (davomi)

Imlo amallar	Qiymat berish va shartli amallar	Tipli amallar	Adresli amallar
() – doirali qavs	= - oddiy qiymar berish	(tip) – tipni o’zgartirish	& - adresni aniqlash
[] – kavadrat qavs	op= - murakkab qiymat berish	sizeof- hajmni hisoblash	* - adres bo’yicha qiymat aniqlash yoki joylash
, - vergul	? – shartli amal		

Arifmetik amallar. Amallar odatda unar ya’ni bitta operandga qo’llaniladigan amallarga va binar ya’ni ikki operandga qo’llaniladigan amallarga ajratiladi.

Binar amallar additiv ya’ni + qo’shuv va – ayirish amallariga , hamda multiplikativ ya’ni * kupaytirish, / bulish va % modul olish amallariga ajratiladi.

Additiv amallarining ustivorligi multiplikativ amallarining ustivorligidan pastroqdir.

Butun sonni butun songa bo’lganda natija butun songacha yahlitlanadi. Misol uchun 20/3=6; (-20)/3=-6; 20/(-3)=-6.

Modul amali butun sonni butun songa bulishdan hosil buladigan qoldikka tengdir. Agar modul amali musbat operandlarga qo’llanilsa, natija ham musbat bo’ladi, aks holda natija ishorasi kompilyatorga bog’likdir.

DASTUR IJRO STRUKTURALARI

Asosan dasturdagi ifodalar ketma-ket, navbatiga ko'ra ijro etiladi.Gohida bir shart bajarilishiga ko'ra, ijro boshqa bir ifodaga o'tadi.Navbatdagi emas, dasturning boshqa yerida joylashgan ifoda bajariladi.Yani sakrash yoki ijro ko'chishi vujudga keladi.60-chi yillarga kelib, dasturlardagi ko'pchilik hatolar aynan shu ijro ko'chishlarining rejasiz ishlatilishidan kelib chiqishi ma'lum bo'ldi.Bunda eng katta aybdor deb bu ko'shishlarni amalga oshiruvchi goto (...ga bor) ifodasi belgilandi.goto dastur ijrosini deyarli istalgan yerga ko'chirib yuborishi mumkin. Bu esa programmani o'qishni va uning strukturasini murakkablashtirib yuboradi. Shu sababli "strukturali dasturlash" atamasi " goto ni yo'q qilish" bilan tenglashtirilardi. Shuni aytib o'tish kerakki, goto kabi shartsiz sakrash amallarini bajaruvchi ifodalar boshqa dasturlash tillarida ham bor.

Tadqiqotlar shuni ko'rsatdiki, istalgan programma goto siz yozilishi mumkin ekan.goto siz yozish uslubi strukturali dasturlash deb nom oldi. Va bunday dastur yozish metodi katta iqtisodiy samara beradi. Strukturali dasturlash asosi shundan iboratki, har bir programma faqatgina uch hil boshqaruv strukturalaridan iboratdir.Bular ifodalarni ketma-ket ijro etish strukturasi (sequence structure), tanlash strukturasi (selection structure) va amalni qayta ijro etish strukturasidir (repetition structure).Ifodalarni ketma-ket ijro etish strukturasi C++ tomonidan ta'minlanadi. Normal sharoitda C++ ifodalari dasturdagi navbatiga ko'ra bajariladi.

BOSHQARUV IFODALARI

Tanlash buyruqlari uchtadir. Bular if, if/else va switch dir. Qayta ijro etish buyruqlari gurugiga ham uchta a'zo bor, bular while, do/while va for. Bularni har birini keyinroq tahlil qilib chiqamiz.Yuqoridagi buyruqlar nomlari C++ dasturlash tilining mahsus so'zlaridir.Dasturchi bu so'zlarni o'zgaruvchi yoki funksiyalar nomi sifatida qo'llashi ta'qiqlanadi.Quyida C++ ning ajratilgan so'zlarining to'liq ro'yhati berilgan.

C++ va C ga tegishli:

auto do goto signed unsigned

break double if sizeof void

case else int static volatile

char enum long struct while

const extern register switch

continue float return typedef

default for short union

Faqat C++ ga qarashli:

asm explicit operator this virtual

bool false private throw wchar_t

catch friend protected true

class inline public try

const_cast mutable reinterpret_cast typeid

delete namespace static_cast typename

dynamic_cast new template using

C++ dagi yetita boshqaruv strukturasini aytib o'tdik. Ular bittagina boshlanish nuqtasiga va bittagina chiqish nuqtasiga egadirlar. Demak biz bu dastur bo'laklarini ketma-ket ulab ketishimiz mumkin.Boshqaruv strukturalarining bu kabi ulanishini devorning g'ishtlarini ustma-ust qalashga ham taqqoslasak bo'ladi.Yoki biz bu bloklarni bir-birining ichiga joylashtirishimiz mumkin.Bu kabi qo'llashish ikkinchi uslub bo'ladi. Mana shu ikki yo'l bilan bog'langan yetita blok yordamida biz istalgan dasturimizni yoza olamiz.

if STRUKTURASI

Biz shartga ko'ra bir necha harakat yo'lidan bittasini tanlaymiz.Misol uchun agar bolaning yoshi 7 ga teng yoki katta bo'lsa u maktabga borishi mumkin bo'lsin. Buni C++ da if ni qo'llab yozaylik.

```
if (yosh >= 7)
maktab();
```

Bu yerda shart bajarilishi yoki bajarilmasligi mumkin. Agar yosh o'zgaruvchisi 7 ga teng yoki undan katta bo'lsa shart bajariladi va maktab() funksiyasi chaqiriladi. Bu holat true (to'g'ri) deyiladi. Agar yosh 7 dan kichik bo'lsa, maktab() tashlab o'tiladi. Yani false (noto'g'ri) holat yuzaga keladi. Biz shart qismini mantiqiy operatorlarga asoslanganligini ko'rib chiqqan edik. Aslida esa shartdagi ifodaning ko'rinishi muhim emas – agar ifodani nolga keltirish mumkin bo'lsa false bo'ladi, noldan farqli javob bo'lsa, musbatmi, manfiymi, true holat paydo bo'ladi va shart bajariladi. Bunga qo'shimcha qilib o'tish kerakki, C++ da mahsus bool tipi mavjud.Bu tipdagi o'zgaruvchilarning yordamida bul (mantiqiy) arifmetikasini amalga oshirish mumkin.bool o'zgaruvchilar faqat true yoki false qiymatlarini olishlari mumkin.

if/else STRUKTURASI

if ni qo'llaganimizda ifoda faqat shart haqiqat bo'lgandagina bajariladi, aks holda tashlanib o'tiladi. if/else yordamida esa shart bajarilmaganda (false natija chiqqanda) else orqali boshqa bir yo'ldan borishni belgilash mumkin. Misolimizni takomillashtirsak.Bola 7 yosh yoki undan katta bo'lsa maktabga, 7 dan kichkina bo'lsa bog'chaga borsin.

```
if (yosh >= 7)
maktab(); //nuqta-vergul majburiydir
else
bogcha();
```

Yuqorida if ga tegishli bo'lgan blok bitta ifodadan (maktab()) iborat. Shu sababli nuqta-vergul qo'yilishi shart.Buni aytib o'tishimizning sababi, masal Pascalda hech narsa qo'yilmasligi shart.C++ da bitta ifosa turgan joyga ifodalar guruhini {} qavslarda olingan holda qo'ysa bo'ladi. Masalan:

```
if (yosh >= 7){
cout<< "Maktabga!\n";
maktab();
}
else{
cout<< "Bog'chaga!\n";
bogcha();
}
```

Aslida har doim {} qavslarni qo'yish yahshi odat hisoblanadi; keyinchalik bir ifoda turgan joyga qo'shimcha qilinganda qavslardan biri unutilib qolmaydi. Strukrurali dasturlashning yana bir harakterli joyi shundaki tabulyatsiya, bo'sh joy va yangi satrlar ko'p qo'llaniladi. Bu programmani o'qishni osonlashtirish uchun qilinadi.C++ uchun bo'sh joyning hech ahamiyati yo'q, lekin dasturni tahrir qilayatgan odamga buyruqlar guruhini, bloklarni tabulyatsiya yordamida ajratib bersak, unga katta yordam bo'ladi. Yuqoridagini quyidagicha ham yozish mumkin:

```
if(yosh>=7){cout<<"Maktabga!\n";maktab()}else{cout<<"Bog'chaga!\n";bog
cha()};
```

Biroq buni o'qish ancha murakkab ishdir. C++ da if/else strukturasiga o'hshash ?: shart operatori (conditional operator) ham bordir. Bu C++ ning bittagina uchta argument oluvchi operatori. Uch operand va shart operatori shart ifodasini beradi. Birinchi operand orqali shartimizni beramiz. Ikkinchi argument shart true (haqiqat) bo'lib chiqqandagi butun shart ifodasining javob qiymatidir.

Uchinchi operand shartimiz bajarilmay (false) qolgandagi butun shart ifodasining qiymatidir. Masalan:

```
bool bayroq;
int yosh = 10;
bayroq = ( yosh >= 7 ? true : false );
```

Agar yosh 7 ga teng yoki katta bo'lsa, bool tipidagi o'zgaruvchimiz true qiymatini oladi, aks taqdirda false bo'ladi. Shart operatori qavslar ichida bo'lishi zarur, chunki uning kuchi katta emas. Javob qiymatlar bajariladigan funksiyalar ham bo'lishi mumkin:

```
yosh>= 7 ? maktab() : bogcha();
```

if/else strukturalarini bir-birining ichida yozishimiz mumkin. Bunda ular bir-biriga ulanib ketadi.Misol uchun tezlikning kattaligiga qarab jarimani belgilab beruvchi blokni yozaylik.

```
if (tezlik > 120)
cout<< "Jarima 100 so'm";
else if (tezlik > 100)
cout<< "Jarima  70 so'm";
else if (tezlik > 85)
cout<< "Jarima  30 so'm";
else
cout<< "Tezlik  normada";
```

Agar tezlik 120 dan katta bo'lsa birinchi if/else strukturasining haqiqat sharti bajariladi.Va bu holda albatta tezlik o'zgaruvchimizning qiymati ikkinchi va uchinchi if/else imizni ham qoniqtiradi. Lekin solishtirish ulargacha bormaydi, chunki ular birinchi if/else ning else qismida, yani noto'g'ri javob qismida joylashgandir. Solishtirish birinchi if/else da tugashi (aynan shu misolda) tanlash amalini tezlashtiradi. Yani bir-biriga bog'liq if/else lar alohida if strukturalari blokidan tezroq bajarilishi mumkin, chunki birinchi holda if/else blokidan vaqtliroq chiqish imkoni bor. Shu sababli ich-ichiga kirgan if/else lar guruhida true bo'lish imkoni ko'proq bo'lgan shartlarni oldinroq tekshirish kerak.

MANTIQIY OPERATORLAR

C++ bir necha solishtirish operatorlariga ega.

Algebraik ifoda	C++ dagi operator	C++ dagi ifoda	Algebraik ma'nosi
tenglik guruhi			
=	==	x==y	x tengdir y ga
teng emas	!=	x!=y	x teng emas y ga
solishtirish guruhi			
>	>	x>y	x katta y dan
<	<	x<y	x kichkina y dan
katta-teng	>=	x>=y	x katta yoki teng y ga
kichik-teng	<=	x<=y	x kichik yoki teng y ga

==, !=, >= va <= operatorlarni yozganda oraga bo'sh joy qo'yib ketish sintaksis hatodir. Yani kompilyator dasturdagi hatoni ko'rsatib beradi va uni tuzatilishini talab qiladi. Ushbu ikki belgili operatorlarning belgilarining joyini almashtirish, masalan <= ni =< qilib yozish ko'p hollarda sintaksis hatolarga olib keladi. Gohida esa != ni =! deb yozganda sintaksis hato vujudga ham, bu mantiqiy hato bo'ladi. Mantiqiy hatolarni kompilyator topa olmaydi.Lekin ular programma ishlash mantig'ini o'zgartirib yuboradi.Bu kabi hatolarni topish esa ancha mashaqqatli ishdir (! operatori mantiqiy inkordir).Yana boshqa hatolardan biri tenglik operatori (==) va tenglashtirish, qiymat berish operatorlarini (=) bir-biri bilan almashtirib qo'yishdir. Bu ham juda ayanchli oqibatlarga olib keladi, chunki ushbu hato aksariyat hollarda mantiq hatolariga olib keladi.

Yuqoridagi solishtirish operatorlarini ishlatadigan bir dasturni ko'raylik.

```
//Mantiqiy solishtirish operatorlari

  # include <iostream.h>

int main()
{
int s1, s2;

cout<< "Ikki son kiriting: " << endl;
cin>> s1 >> s2;                    //Ikki son olindi.

if (s1 == s2) cout << s1 << " teng " << s2 << " ga" << endl;
if (s1 < s2) cout << s1 << " kichik " << s2 << " dan" << endl;
if (s1 >= s2) cout << s1 << " katta yoki teng " << s2 << " ga" << endl;
if (s1 != s2) cout << s1 << " teng emas " << s2 << " ga" << endl;

return (0);
}
```

Ekranda:

```
Ikki sonni kiriting: 74 33
74 katta yoki teng 33 ga
```

74 teng emas 33 ga

Bu yerda bizga yangi bu C++ ning if (agar) struktura-sidir. if ifodasi ma'lum bir shartning to'g'ri (true) yoki noto'g'ri (false)bo'lishiga qarab, dasturning u yoki bu blokini bajarishga imkon beradi. Agar shart to'g'ri bo'lsa, if dan so'ng keluvchi amal bajariladi. Agar shart bajarilmasa, u holda if tanasidagi ifoda bajarilmay, if dan so'ng keluvchi ifodalar ijrosi davom ettiriladi. Bu strukturaning ko'rinishi quyidagichadir:

if (shart) ifoda;

Shart qismi qavs ichida bo'lishi majburiydir.Eng ohirida keluvchi nuqta-vergul (;) shart qismidan keyin qo'yilsa (if (shart); ifoda;) mantiq hatosi vujudga keladi. Chunki bunda if tanasi bo'sh qoladi. Ifoda qismi esa shartning to'g'ri-noto'g'ri bo'lishiga qaramay ijro qilaveradi.

C++ da bitta ifodani qo'yish mumkin bo'lgan joyga ifodalar guruhini ham qo'yish mumkin.Bu guruhni {} qavslar ichida yozish kerak.if da bu bunday bo'ladi:

```
if (shart) {
  ifoda1;
  ifoda2;
  ...
ifodaN;
}
```

Agar shart to'g'ri javobni bersa, ifodalar guruhi bajariladi, aksi taqdirda blokni yopuvchi qavslardan keyingi ifodalardan dastur ijrosi davom ettiriladi.

Boshqaruv strukturalarida shart qismi bor dedik.Shu paytgacha ishlatganshartlarimiz ancha sodda edi.Agar bir necha shartni tekshirmoqchibo'lganimizda ayri-ayri shart qismlarini yozardik.Lekin C++ da bir nechasodda shartni birlashtirib, bitta murakkab shart ifodasini tuzishga yordamberadigan mantiqiy operatorlar mavjuddir. Bilar mantiqiy VA - && (AND),mantiqiy YOKI - || (OR) va mantiqiy INKOR - ! (NOT). Bular bilan misol keltiraylik.

Faraz qilaylik, bir amalni bajarishdan oldin, ikkala shartimiz (ikkitadan

ko'p ham bo'lishi mumkin) true (haqiqat) bo'lsin.

if (i < 10 && l >= 20){...}

Bu yerda {} qavslardagi ifodalar bloki faqat i 10 dan kichkina va l 20 dan katta yoki teng bo'lgandagina ijro ko'radi.

AND ning (&&) jadvali:

ifoda1 ifoda2 ifoda1 && ifoda2

false (0) false (0) false (0)

true (1) false (0) false (0)

false (0) true (1) false (0)

true (1) true (1) true (1)

Bu yerda true ni yeriga 1, false ni qiymati o'rniga 0 ni qo'llashimiz mumkin.

Boshqa misol:

while (g<10 || f<4){

...

}

Bizda ikki o'zgaruvchi bor (g va f). Birnchisi 10 dan kichkina yoki ikkinchisi

4 dan kichkina bo'lganda while ning tanasi takrorlanaveradi. Yani shart bajarilishi uchun eng kamida bitta true bo'lishi kerak, AND da (&&) esa hamma oddiy shartklar true bo'lishi kerak.

OR ning (||) jadvali:

ifoda1 ifoda2 ifoda1 || ifoda2

false (0) false (0) false (0)

true (1) false (0) true (1)

false (0) true (1) true (1)

true (1) true (1) true (1)

&&va || operatorlari ikkita argument olishadi. Bulardan farqli o'laroq, ! (mantiqiy inkor) operatori bitta argumet oladi, va bu argumentidan oldin qo'yiladi. Inkor operatori ifodaning mantiqiy qiymatini teskarisigao'zgartiradi. Yani false ni true deb beradi, true ni esa false deydi.

Misol uchun:

```
if ( !(counter == finish) )
cout<< student_bahosi << endl;
```

Agar counter o'zgaruvchimiz finish ga teng bo'lsa, true bo'ladi, bu true qiymat esa !yordamida false ga aylanadi. false qiymatni olgan if esa ifodasini bajarmaydi. Demak ifoda bajarilishi uchun bizga counter finish ga teng bo'lmagan holati kerak. Bu yerda !ga tegishli ifoda () qavslar ichida bo'lishi kerak. Chunki mantiqiy operatorlar tenglilik operatorlaridan kuchliroqdir. Ko'p hollarda !operatori o'rniga mos keladigan mantiqiy tenglilik yoki solishtirish operatorlarini ishlatsa bo'ladi, masalan yuqoridagi misol quyidagi ko'rinishda bo'ladi:

```
if (counter != finish)
cout<< student_bahosi << endl;
```

NOT ning jadvali:

ifoda !(ifoda)

false (0) true (1)

true (1) false (0)

for TAKRORLASH STRUKTURASI

for strukturasi sanovchi (counter) bilan bajariladigan takrorlashni bajaradi.Boshqa takrorlash bloklarida (while, do/while) takrorlash sonini control qilish uchun ham sanovchini qo'llasa bo'lardi, bu holda takrorlanish sonini o'ldindan bilsa bo'lardi, ham boshqa bir holatning vujudga kelish-kelmasligi orqali boshqarish mumkin edi. Ikkinchi holda ehtimol miqdori katta bo'ladi.

Masalan qo'llanuvchi belgilangan sonni kiritmaguncha takrorlashni bajarish kerak bo'lsa biz while li ifodalarni ishlatamiz. for da esa sanovchi ifodaning qiymati oshirilib (kamaytirilib) borilvuradi, va chegaraviy qiymatni olganda takrorlanish tugatiladi. for ifodasidan keyingi bitta ifoda qaytariladi. Agar bir necha ifoda takrorlanishi kerak bo'lsa, ifodalar bloki

{} qavs ichiga olinadi.

```
//Ekranda o'zgaruvching qiymatini yozuvchi dastur, for ni ishlatadi.
# include <iostream.h>
int main()
{
for (int i = 0; i < 5; i++){
cout<< i << endl;
 }
return (0);
}
```

Ekranda:

```
0
1
2
```

3

4

for strukturasi uch qismdan iboratdir. Ular nuqta-vergul bilan bir-biridan ajratiladi.for ning ko'rinishi:

for(1. qism ; 2. qism ; 3. qism){

takror etiladigan blok

}

1. qism - e'lon va initsalizatsiya.

2. qism - shartni tekshirish (oz'garuvchini chegaraviy

qiymat bilan solishtirish).

3.qism - o'zgaruvchining qiymatini o'zgartirish.

Qismlarning bajarilish ketma-ketligi quyidagichadir:

Boshida 1.qism bajariladi (faqat bir marta), keyin 2. qismdagi shart tekshiriladi va agar u true bo'lsa takrorlanish bloki ijro ko'radi, va eng oxirida 3. qismda o'zgaruvchilar o'zgartiriladi, keyin yana ikkinchi qismga o'tiladi. for strukturamizni while struktura bilan almashtirib ko'raylik:

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

cout<< "Hello!"<<endl;
```

Ekranga 10 marta Hello! so'zi bosib chiqariladi. I o'zgaruvchisi 0 dan 9 gacha o'zgaradi.i 10 ga teng bo'lganda esa i < 10 sharti noto'g'ri (false) bo'libchiqadi va for strukturasi nihoyasiga yetadi. Buni while bilan yozsak:

```
 int i = 0;
```

```
while ( i<10 ){

cout<< "Hello!" <<endl;

i++;

}
```

Endi for ni tashkil etuvchi uchta qismninig har birini alohida ko'rib chiqsak. Birinchi qismda asosan takrorlashni boshqaradigan sanovchi (counter) o'zgaruvchilar e'lon qilinadi va ularga boshlangich qiymatlar beriladi (initsalizatsiya). Yuqoridagi dastur misolida buni int i = 0; deb berganmiz. Ushbu qismda bir necha o'zgaruvchilarni e'lon qilishimiz mumkin, ular vergul bilan ajratilinadi.Ayni shu kabi uchinchi qismda ham bir nechta o'zgaruvchilarning qiymatini o'zgartirishimiz mumkin.Undan tashqari birinchi qismda for dan oldin e'lon qilingan o'zgaruvchilarni qo'llasak bo'ladi.

Masalan:

```
 int k = 10;

int l;

for (int m = 2, l = 0 ; k <= 30 ; k++, l++, ++m) {

cout<< k + m + l;

}
```

Albatta bu ancha sun'iy misol, lekin u bizga for ifodasining naqadar moslashuvchanligi ko'rsatadi.for ning qismlari tushurib qoldirilishi mumkin.

Masalan:

```
for(;;) {}
```

ifodasi cheksiz marta qaytariladi. Bu for dan chiqish uchun break operatorini beramiz. Yoki agar sanovchi sonni takrorlanish bloki ichida o'zgartirsak, for ning 3.qismi kerak emas. Misol:

```
for(int g = 0; g < 10; ){
cout<< g;
g++;
}
```

Yana qo'shimcha misollar beraylik.

```
for (int y = 100; y >= 0; y-=5){
...
ifoda(lar);
...
}
```

Bu yerda 100 dan 0 gacha 5 lik qadam bilan tushiladi.

```
for(int d = -30; d<=30; d++){
...
ifoda(lar);
...
```

} 60 marta qaytariladi.

for strukrurasi bilan dasturlarimizda yanada yaqinroq tanishamiz. Endi 1.qismda e'lon qilinadigan o'zgaruvchilarning hususiyati haqida bir og'iz aytib o'taylik. Standartga ko'ra bu qismda e'lon qilingan o'zgaruvchilarning qo'llanilish sohasi faqat o'sha for strukturasi bilan chegaralanadi.Yani bitta blokda joylashgan for strukturalari mavjud bo'lsa, ular ayni ismli o'zgaruvchilarni qo'llana ololmaydilar. Masalan quyidagi hatodir:

```
for(int j = 0; j<20 ; j++){...}
```

...

for(int j = 1; j<10 ; j++){...} //hato!

j o'zgaruvchisi birinchi for da e'lon qilinib bo'lindi. Ikkinchi for da ishlatish mumkin emas.Bu masalani yechish uchun ikki hil yo'l tutish mumkin.Birinchisi bitta blokda berilgan for larning har birida farqli o'zgaruvchilarni qo'llashdir. Ikkinchi yo'l for lar guruhidan oldin sanovchi vazifasini bajaruvchi bir o'zgaruvchini e'lon qilishdir. Va for larda bu o'zgaruvchiga faqat kerakli boshlang’ich qiymat beriladi halos.

for ning ko'rinishlaridan biri, bo'sh tanali for dir.

for(int i = 0 ; i < 1000 ; i++) Buning yordamida biz dastur ishlashini sekinlashtirishimiz mumkin.

Dasturiy qism

Masalaning qo’yilishi:

Kiritilgan N sonini bo’luvchilarini chiqaruvchi dastur.

Masalani matematik tahlili:

Masalada ixtiyoriy N sonini bo’luvchilarini topish so’ralmoqda. Biz bilamizki biror bir sonning bo’luvchilari deganda uni shu songa qoldiqsiz bo’linishi tushuniladi.Demak, ixtiyoriy N sonini bo’luvchilarini topish uchun uning o’zidan kichik bo’lgan barcha natural sonlarga bo’lib chiqamiz va ulardan qoldiqsiz bo’linma hosil qilganlarini bo’luvchi deb qabul qilamiz.

Masalan:

N=12

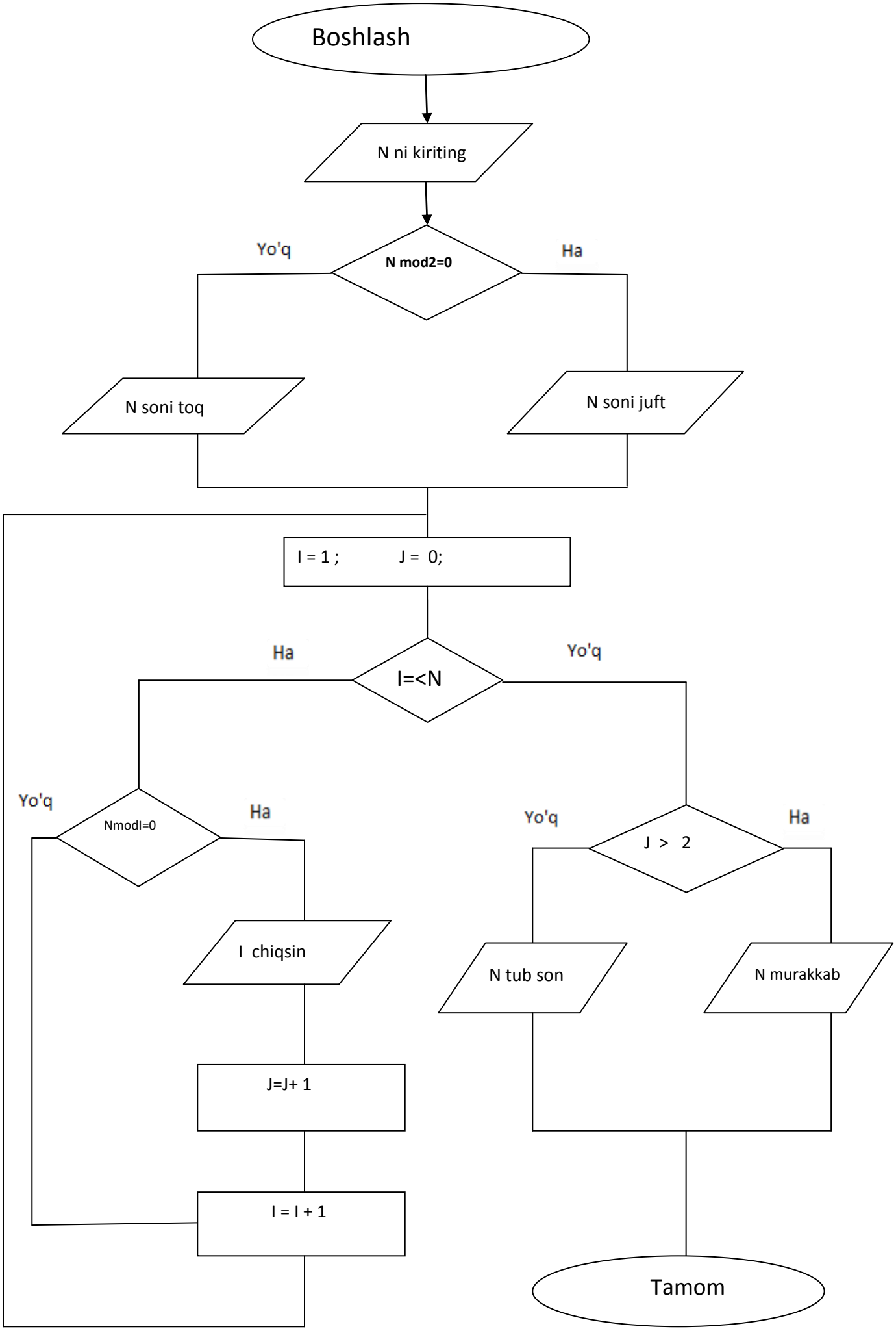
Uning o’zidan kichik bo’lgan barcha natural sonlar 1,2,3,4,5,6,7,8,9,10,11,12

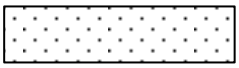
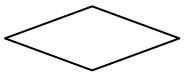
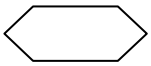
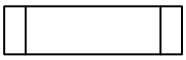
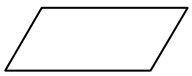
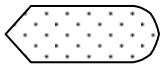
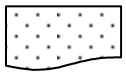
Endi ularni har biri bilan tekshirib chiqamiz.

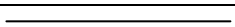
- 12/1=12 qoldiq 0
- 12/2=6 qoldiq 0
- 12/3=4 qoldiq 0
- 12/4=3 qoldiq 0
- 12/5=2 qoldiq 2
- 12/6=2 qoldiq 0
- 12/7=1 qoldiq 5
- 12/8=1 qoldiq 4
- 12/9=1qoldiq 3
- 12/10=1 qoldiq 2
- 12/11=1 qoldiq 1
- 12/12=1 qoldiq 0

Demak bu sonimizning bo’luvchilari 1,2,3,4,6,12 ekan . Chunki qolgan 5,7,8,9,10,11 sonlari qoldiqli bo’lindi. 12 sonimizning bo’luvchilari 6 ta ekan ,u son murakkab va juft ekan.

Dasturlashda masalaning matematik tahlilidan kelib chiqib uning blok sxemasi chizib olinadi.Masalaning blok sxemasi quyidagicha


Boshlash
N ni kiriting
Yo'q
N mod2=0
Ha
N soni toq
N soni juft
I = 1 ;
J = 0;
Ha
I=<N
Yo'q
Yo'q
NmodI=0
Ha
I chiqsin
J=J+ 1
I = I + 1
Yo'q
J > 2
Ha
N tub son
N murakkab
Tamom

Nomi	Belgilanishi	Bajaradigan vazifasi
Jarayon		Bir yoki bir nechta amallarni bajarilishi natijasida maghlu-motning qiymati yoki shaklini ohzgartirish
qaror		Biron bir shartga boghlik ravishda algoritmni bajarilish yohnalishini tanlash
Shakl ohzgartirish		Dasturni ohzgartiruvchi buyruq yoki buyruqlar turkumini ohz-gartirish amalini bajarish
Avval aniqlan-gan jarayon		Oldindan ishlab chiqilgan dastur yoki algoritmdan foydalanish
Kiritish-chiqarish		Axborotlarni qayta ishlash mumkin bohlgan shaklga ohtkazish (kiritish) yoki olingan natijalarni tasvirlash (chiqarish)
Displey		EOhMga ulangan displeydan axborotlarni kiritish yoki chiqarish
Ohujjat		Axborotlarni qoghozga chiqarish yoki qoghozdan kiritish

Axborotlar oqimi chizighi		Bloklarlar orasidagi boghlanishlarni tasvirlash
Boghlagich		Uzilib qolgan axborot oqimlarini ulash belgisi
Boshlash – tugatish		Axborotniqaytaishlashniboshlash, vaqtinchatohxtatishyokitohxtatibqohyish

Yo’naltiruvchichiziq, blok-sxemadagiharakatningboshqaruvinibelgilaydi.

Blok-sxemaichidaohisoblashlarningtegishlibosqichlariko’rsatiladi.Shuyerdaharbirsimvolbatafsiltushuntiriladi.

Har bir blok o’z raqamiga ega bo’ladi. U tepadagi chap burchakka chiziqni uzib yozib qo’yiladi. Blok-sxemadagi grafik simvollar hisoblash jarayonining rivojlanish yo’nalishini ko’rsatuvchi chiziqlar bilan birlashtiriladi.Ba’zan chiziqlar oldida ushbu yo’nalish qanday sharoitda tanlanganligi yozib qo’yiladi. Axborot oqimining asosiy yo’nalishi tepadan pastga va chapdan o’ngga ketadi. Bu hollarda chiziqlarni ko’rsatmasa ham bo’ladi, boshqa hollarda albatta chiziqlarni qo’llash majburiydir. Blokka nisbatan oqim chizig’i kiruvchi yoki chiquvchi bo’lishi mumkin. Blok uchun kiruvchi chiziqlar soni chegaralanmagan.Chiquvchi chiziq esa mantiqiy bloklardan boshqa hollarda faqat bitta bo’ladi. Mantiqiy bloklar ikki va undan ortik oqim chizig’iga ega bo’ladi. Ulardan har biri mantiqiy shart tekshirishining mumkin bo’lgan natijalarga mos keladi.

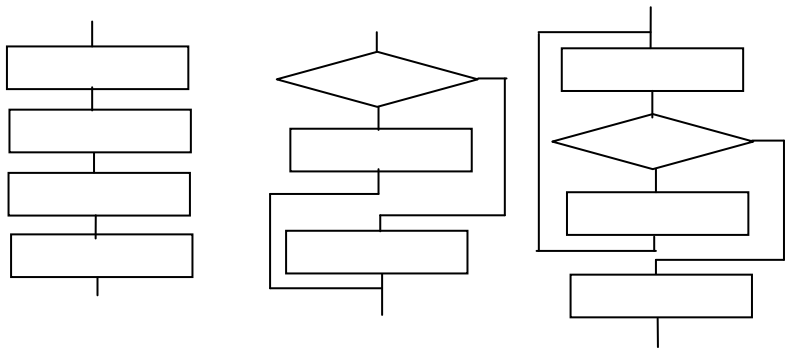
O’zaro kesishadigan chiziqlar soni ko’p bo’lganda va yo’nalishlari ko’p o’zgarganda tuzimdagi ko’rgazmalik yo’qoladi. Bunday hollarda axborot oqimi chizig’i uzishga yo’l qo’yiladi, uzilgan chiziq uchlariga "birlashtiruvchi" belgisi qo’yiladi.Agar uzilish bitta sahifa ichida bo’lsa, O belgisi ishlatilib, ichiga ikki

tarafga ham bir xil harf-raqam belgisi qo’yiladi.Agar tizim bir necha sahifaga joylansa, bir sahifadan boshqasiga o’tish "sahifalararo bog’lanish" belgisi ishlatiladi. Bunda axborot uzatilayotgan sahifadagi blokga qaysi sahifa va blokka borishi yoziladi, qabul qilinayotgan sahifada esa qaysi sahifa va blokdan kelishi yoziladi.

Algoritmlarning turlari

Algoritmlarni asosan 3 turga bo’lish mumkin:

1) Chiziqli algoritmlar ;
2) Tarmoqlanuvchi algoritmlar;
3) Takrorlanuvchi algoritmlar.



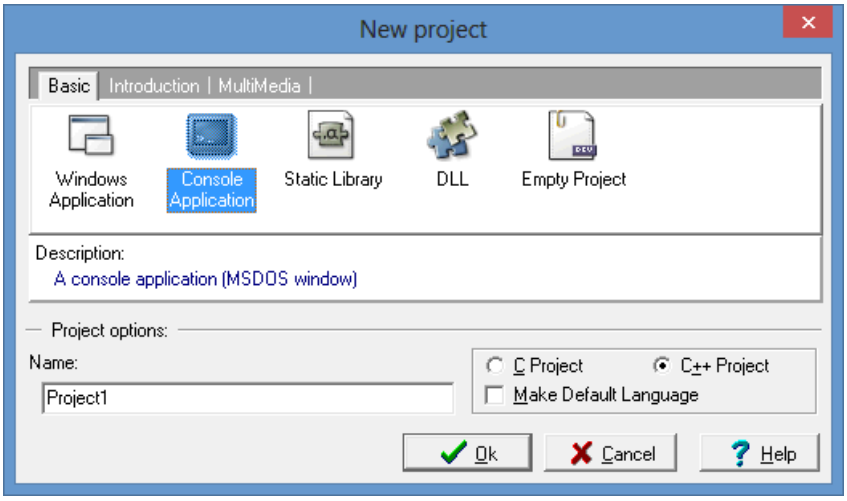
1) 2) 3)

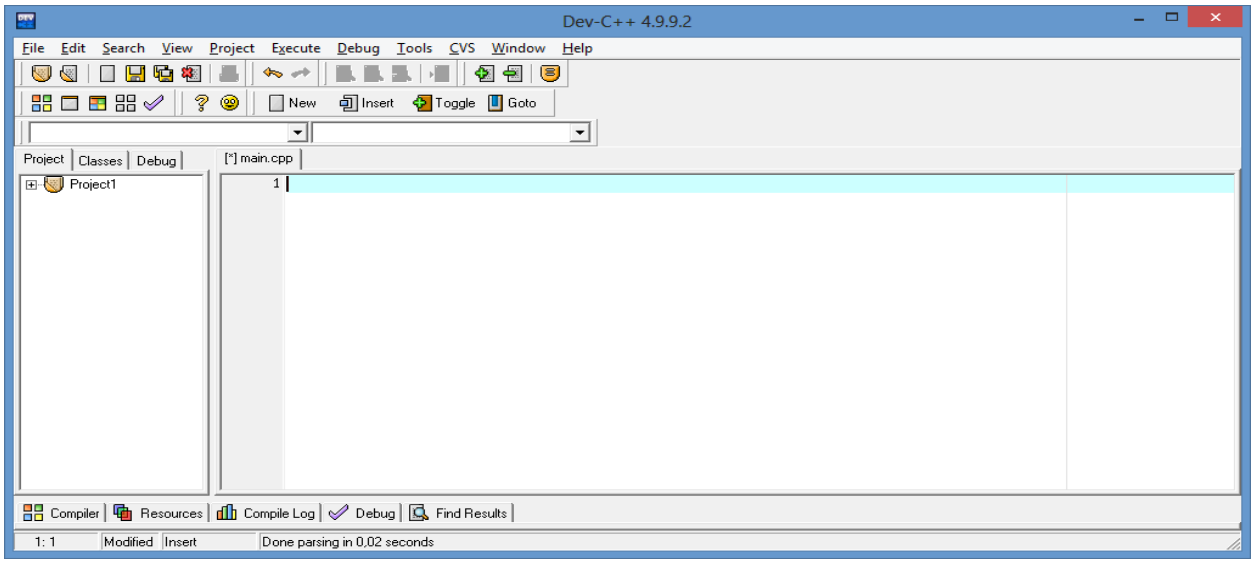
Demak bizning masalaning algoritmi takrorlanuvchi algoritmga misol bo’larkan.

Masalaning Dev C++ dasturlash muhitida ishlash jarayoni

Eng avvalo Dev C++ dasturlash muhitini quyidagicha ishga tushiramiz.

Dastur ishga tushgandan so’ng New bo’limidan quyidagilarni bajaramiz.

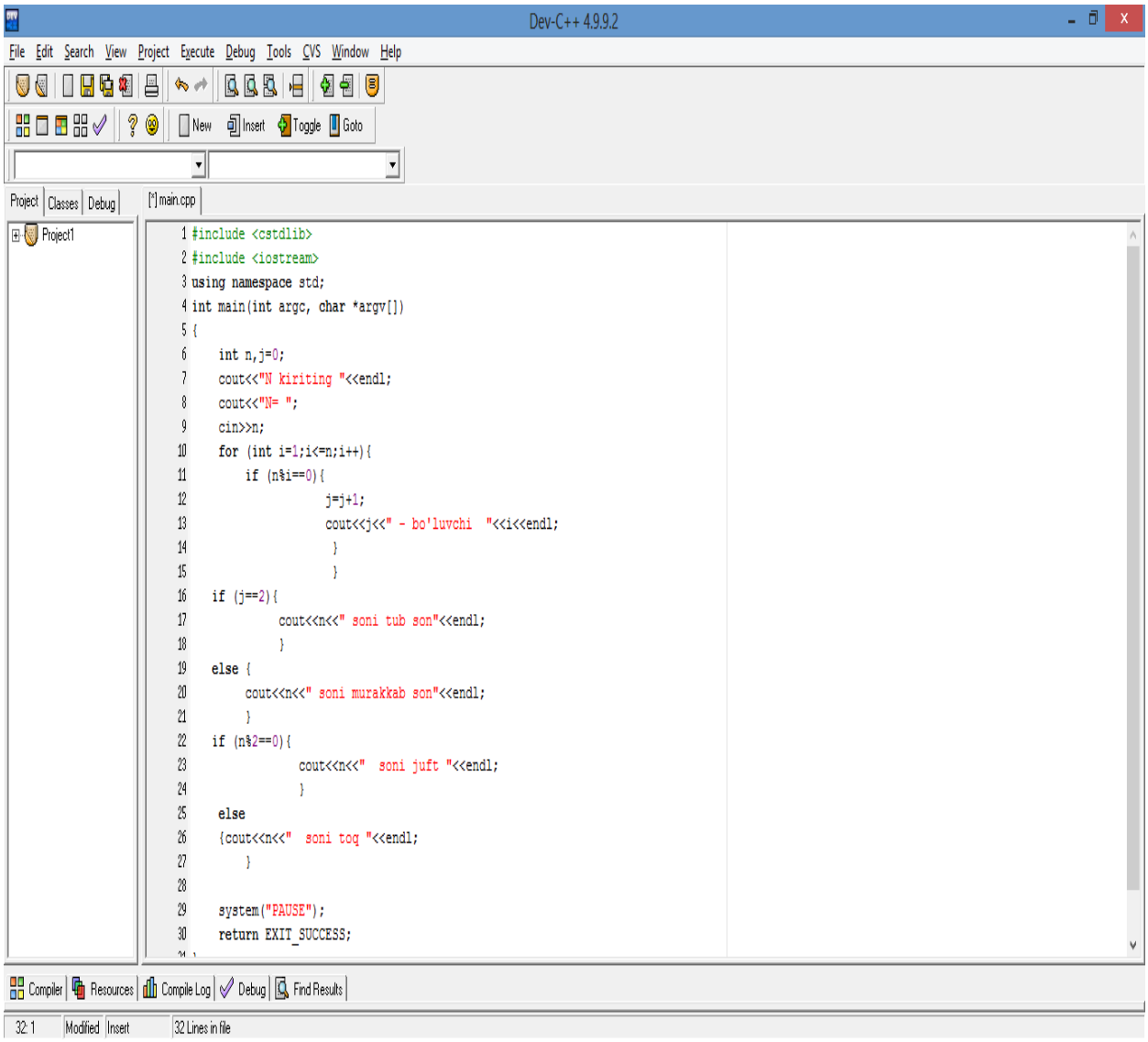




Dasturning kodlar oynasiga quyidagi kodlarni kiritamiz.

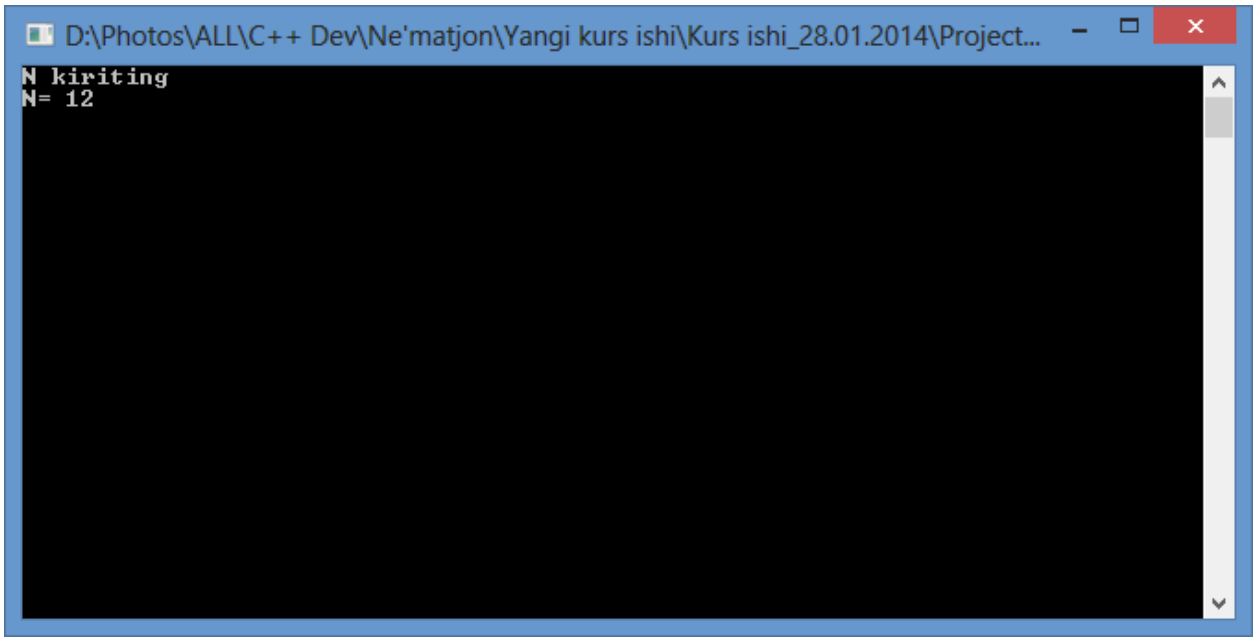
```
#include <cstdlib>
#include <iostream>

using namespace std;
int main(int argc, char *argv[])
{
    int n,j=0;
    cout<<"N kiriting "<<endl;
    cout<<"N= ";
    cin>>n;
    for (int i=1;i<=n;i++)
{
        if (n%i==0){
                j=j+1;
                cout<<j<<" - bo'luvchi  "<<i<<endl;
                 }
                 }
  if (j==2){
           cout<<n<<" soni tub son"<<endl;
           }
  else {
       cout<<n<<" soni murakkab son"<<endl;
       }
  if (n%2==0){
              cout<<n<<"  soni juft "<<endl;
              }
   else
```

```
    {cout<<n<<"  soni toq "<<endl;
        }
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

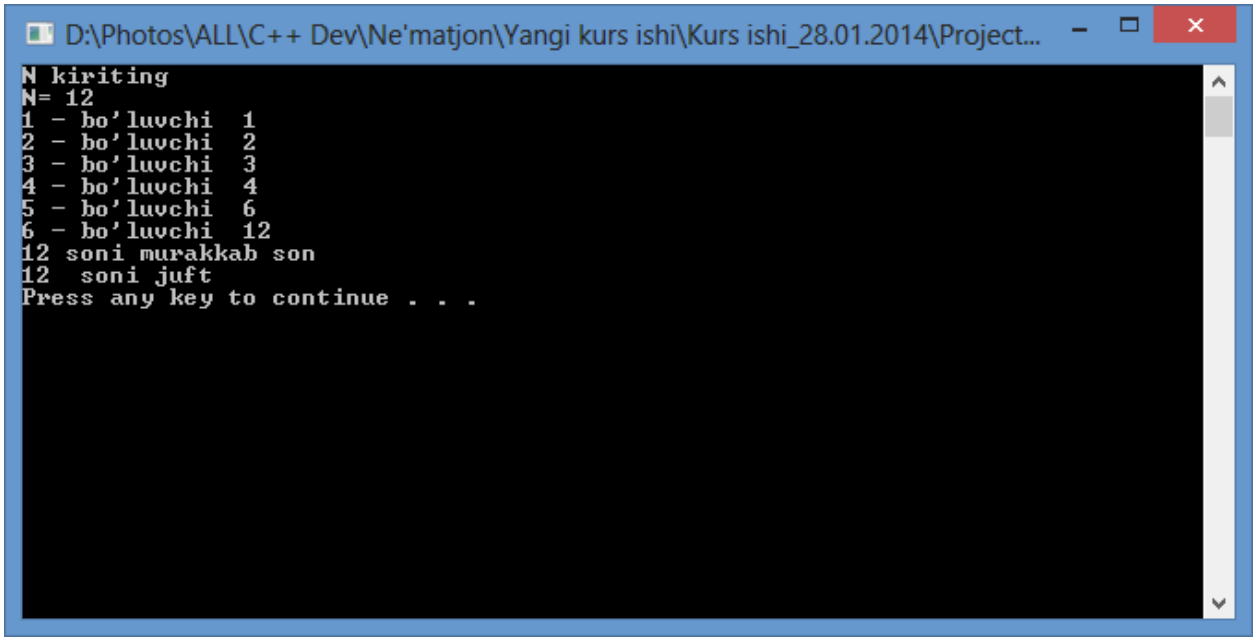


Endi yaratilgan dasturni Ctrl+F9 tugmalari yordamida kompilyatsiya qilamiz va F9 tugmasi bilan ishga tushiramiz.

So’ng N ning qiymatini masalan 12 deb kiritamiz.



Keyin quyidagicha natijaga ega bo’lamiz.



Xulosa

Men ushbu kurs ishimnii bajarish davomida Dev C++ tilining sikllar bilan ishlash imkoniyati va shart operatorlari bilan ishlashni mukkamal darajada o’rganib chiqdim . Dev C++ tilida amaliy dastur yaratish mobaynida shunga amin bo’ldimki, berilgan masalani yechish uchun birinchi navbatda uning mohiyatini to’liq anglab olish kerakligini tushundim. Chunki, masalani tushunib olish masalani yarim yechimi xisoblanishini bilamiz. Dastur tuzishda uning har operatorlaridan kerakli joyda kerakli tartibda foydalanish dasturning hajm jihatdan va sifat ko’rsatkichini yuqori darajada ekanligini ifodalaydi. Eng asosiysi dasturning foydalanuvchi bilan interfeysini yaxshi tashkillashimiz kerakligi o’rgandim.

Foydalanilgan adabiyotlar

1. G’ulomov S. S. va boshqalar. Axborot tizimlari va texnologiyalari: Oliy o‘kuv yurti talabalari uchun darslik Akademik S. S. G’ulomovning umumiy taxriri ostida.—T.: «Sharq», 2000.

2. N.G. Mardanova.Programmalash bo`yichachuqurlashtirilgan kurs T 2006
3. Arslonov K. C++ dan qo’llanma T .:2010
4. Ahunjonov M. C++ tili dasturlash asoslari .Farg’ona – 2009
5. **www.acm.tuit.uz**
6. **www.ziyonet.uz**
7. **www.google.ru**
8. **www.dastur.uz**