O’ZBEKISTON RESPUBLIKASI ALOQA, AXBOROTLASHTIRISH VA TELEKOMMUNIKATSIYA TEXNOLOGIYALARI DAVLAT QO’MITASI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI
NUKUS FILIALI

“Kompyuter injiniringi“ fakulteti

“Kompyuter injiniringi”yo’nalishi

Taqsimlangan algoritmlar va sistemalar

fanidan tayyorlangan

“ Taqsimlangan sistemalar va yo’qori unumli texnologialar ”
mavzusidagi

Kurs ishi

Qabul qilgan: ***Tatlimuratov N***

Bajargan: ***301-13 guruhi*** ***Asanov Sharapatdin***

NUKUS 2017-YIL

Reja:

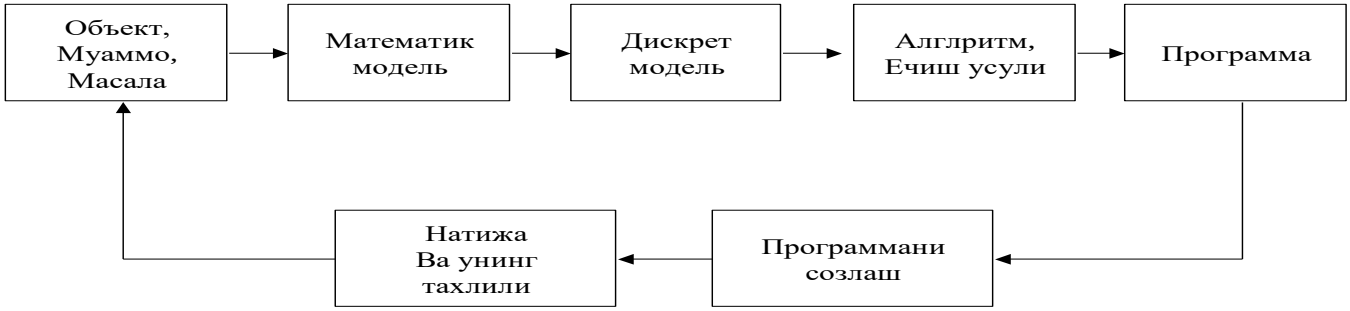
1. **KIRISH**
2. **ASOSIY QISM**
   A) Algoritmning asosiy xossalari
   B) Algoritmning tasvirlash usullari
   C) Algoritm ijrosini tekshirish
3. **XULOSA**
4. **AMALIY ISH**
5. **FOYDALANGAN ADABIYOTLAR**

Kirish

Algoritmlar

Hisoblash eksperimenti.

Odatda tabiat yoki jamiyatda uchraydigan turli muammo, masala yoki jarayonlarni o'rganishni EHM yordamida olib borish uchun, birinchi navbatda, qaralayotgan masala, jarayon - ob'ektning matematik ifodasi, ya'ni matematik modelini ko'rish kerak bo'ladi. Qaralayotgan ob'ektning matematik modelini yaratish juda murakkab jarayon bo'lib, o'rganilayotgan ob'ektga bog'liq ravishda turli soha mutaxassislarining ishtiroki talab etiladi. Umuman, biror masalani EHM yordamida echishni quyidagi bosqichlarga ajratish mumkin.



1-rasm. Hisoblash eksperimentining sxemasi

Misol sifatida, kosmik kemani erdan Zuxro planetasiga eng optimal traektoriya bo'yicha uchirish masalasini xal qilish talab qilingan bo'lsin.

Birinchi navbatda, qo'yilgan masala turli soha mutaxassislari tomonidan atroflicha o'rganilishi va bu jarayonni ifodalaydigan eng muhim - bo'lgan asosiy parametrlarni aniqlash kerak bo'ladi. Masalan, fizik-astronom-injener tomonidan, masala qo'yilishining o'rinli ekanligi, yani planetalar orasidagi masofa va atmosfera qatlamlarining ta'siri, er tortish kuchini engib o'tish va kemaning og'irligi, zarur bo'lgan yoqilg'ining optimal miqdori va kosmik kemani qurishda qanday materiallardan foydalanish zarurligi, inson sog'lig'iga ta'siri va sarflanadigan vaqt va yana turli tuman ta'sirlarni hisobga olgan holda shu masalaning matematik modelini tuzish zarur bo'ladi. Zikr etilgan ta'sirlarni va fizikaning qonunlarini hisobga olgan holda bu masalani ifodalaydigan birorta differentsial yoki boshqa ko'rinishdagi modellovchi tenglama hosil qilish mumkin bo'ladi. Balki, bu masalani bir nechta alohida masalalarga bo'lib o'rganish maqsadga muvofiqdir. Bu matematik modelni o'rganish asosida bu masalani ijobiy echish yoki xozirgi zamon tsiviliziyatsiyasi bu masalani echishga qodir emas degan xulosaga xam kelish mumkin. Bu fikrlar, yuqorida keltirilgan jadvalning 2 blokiga mos keladi.

Faraz qilaylik biz matematik modelni qurdik. Endi uni EHM da echish masalasi tug'iladi. Bizga Ma'lumki, EHM faqat 0 va 1 diskret qiymatlar va ular

ustida arifmetik va mantiqiy amallarni bajara oladi xolos. SHuning uchun matematik modelga mos diskret modelni qurish zaruriyati tug'iladi (1-rasm, 3-blok). Odatda, matematik modellarga mos keluvchi diskret modellar ko'p noma'lumli murakkab chiziqsiz algebraik tenglamalar sistemasi (chekli ayirmali tenglamalar-sxemalar) ko'rinishida bo'ladi(4-blok). Endi hosil bo'lgan diskret modelni sonli echish usulini–algoritmini yaratish zarur bo'ladi. Algoritm esa tuziladigan programma uchun asos bo'ladi. Odatda, tuzilgan programmani ishchi holatga keltirish uchun programmaning xato va kamchiliklarini tuzatish – sozlash zarur bo'ladi. Olingan sonli natijalar hali programmaning to'g'ri ishlayotganligi kafolatini bermaydi. SHuning uchun olingan natijalarni masalaning mohiyatidan kelib chiqqan holda analiz qilish kerak bo'ladi. Agar olingan natija o'rganilayotgan jarayonni ifodalamasa, masalani 1-rasmdagi sxema asosida qaytadan ko'rib chiqish va zarur bo'lgan joylarda o'zgartirishlar kiritish kerak bo'ladi. Bu jarayon, to kutilan ijobiy yoki salbiy natija olinguncha davom ettiriladi va bu takrorlanuvchi jarayonga Hisoblash eksperimenti deb ataladi. Odatda, hisoblash eksperimenti deganda soddaroq holda, model, algoritm va programma uchligini (triadasini) tushunish mumkin.

Algoritm tushunchasi

Yuqorida qayd qilganimizdek, qo'yilgan biror masalani EHMda echish uchun, avval uning matematik modelini, keyin algoritmini va programmasini tuzish kerak bo'ladi. Bu uchlikda algoritm bloki muhim ahamiyatga ega. Endi algoritm tushunchasining ta'rifi va xossalarini bayon qilamiz.

Algoritm bu oldimizga qo'yilgan masalani echish zarur bo'lgan amallar ketma-ketligidir.

Masalan kvadrat tenglamani echish uchun quyidagi amallar ketma-ketligi zarur bo'ladi:

1. a,v,s- koeffiientlar berilgan bo'lsin,
2. berilgan a,b,c- koeffiientlar yordamida diskriminant D=b^2-4ac hisoblanadi,
3. D>0 bo'lsa $X_{1/2} = \left(-b \neq \sqrt{D}\right)/(2*a)$
4. D<0 bo'lsa haqiqiy echim yo'q

Misol sifatida yana berilgan a, v, s tomonlari bo'yicha uchburchakning yuzasini Geron formulasi bo'yicha hisoblash masalasini ko'rib o'taylik.

1. a, v, s –uchburchakning tomonlari uzunliklari,
2. r= (a+v+s)/2 –perimetrning yarmi hisoblansin,
3. T=p(r-a)(r-v)(r-s) hisoblansin,
4. S=$\sqrt{}$ T hisoblansin.

Yuqoridagi misollardan ko’rinib turibdiki, algoritmning xar bir qadamda bajariladigan amallar tushinarli va aniq tarzda ifodalangan, hamda chekli sondagi amallardan keyin aniq natijani olish mumkin.

Fikr etilgan, tushinarlilik, aniqlik, cheklilik va natijaviylik tushunchalari algoritmning asosiy xossalarini tashkil etadi. Bu tushunchalar keyingi pararaflarda alohida ko’rib o’tiladi.

Algoritm so’zi va tushunchasi IX asrda yashab ijod etgan buyuk alloma Muhammad al-Xorazmiy nomi bilan uzviy bog’liq. Algoritm so’zi Al-Xorazmiy nomini Evropa 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.

1. **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.
2. **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 tushinarli 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 tushinmasligi natijasida bajara olmaydi, lekin "son o’zini o’ziga ko’paytirilsin" shaklidagi ko’rsatmani bemalol bajaradi, chunki u ko’rsatma mazmunidan ko’payirish amalini bajarish kerakligini anglaydi.

3. **Aniqlik.** Ijrochiga berilayotgan ko’rsatmalar aniq mazmunda bo’lishi zarur. CHunki ko’rsatmadagi noaniqliklar mo’ljaldagi maqsadga erishishga olib kelmaydi. Odam uchun tushinarli bo’lgan "3-4 marta silkitilsin", "5-10 daqiqa qizdirilsin", "1-2 qoshiq solinsin", "tenglamalardan biri echilsin" 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.

4. **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 algorim shu xildagi har qanday masalani echishga yaroqli bo’lishi kerak. Masalan, ikki oddiy kasrning umumiy mahrajini topish algoritmi, kasrlarni turlicha o’zgartirib bersangiz ham ularning umumiy mahrajlarini aniqlab beraveradi. YOki uchburchanning yuzini topish algoritmi, uchburchakning qanday bo’lishidan qat’iy nazar, uning yuzini hisoblab beraveradi.

5. **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 echimga 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 misollarda odatda biz masalani echish 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.

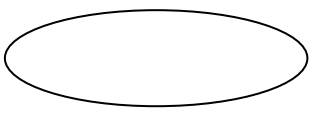
2. **Algoritmning formulalar bilan berilish** usulidan matematika, fizika, kimyo kabi aniq fanlardagi formulalarni o’rganishda foydalaniladi. Bu usulni ba’zan analitik ifodalash deyiladi.

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. Funktsiyalarning grafiklarini chizishda ham algoritmlarning qiymatlari jadvali ko’rinishlaridan foydalanamiz. Bu kabi jadvallardan foydalanish algoritmlari sodda bo’lgan tufayli ularni o’zlashtirib olish oson.

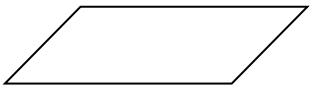
Yuqorida ko’rilgan algoritmlarning tasvirlash usullarining asosiy maqsadi, qo’yilgan masalani echish uchun zarur bo’lgan amallar ketma-ketligining eng qulay holatinni aniqlash va shu bilan odam tomonidan programma yozishni yanada osonlashtirishdan iborat. Aslida programma ham algoritmning boshqa bir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

Blok-sxemalarni tuzishda foydalaniladigan asosiy sodda geometrik figuralar quyidagilardan iborat.



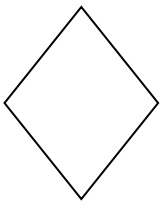
Oval (ellips shaklli), u algoritmning boshlanishi yoki tugallashini belgilaydi.

To’g’ri burchakli to’rtburchak, qiymat berish yoki tegishli ko’rsatmalarni bajarish jarayonini belgilaydi.



Parallelogramm, ma’lumotlarni kiritish yoki chiqarishni belgilaydi.

Yordamchi algoritmga murojatni belgilaydi.



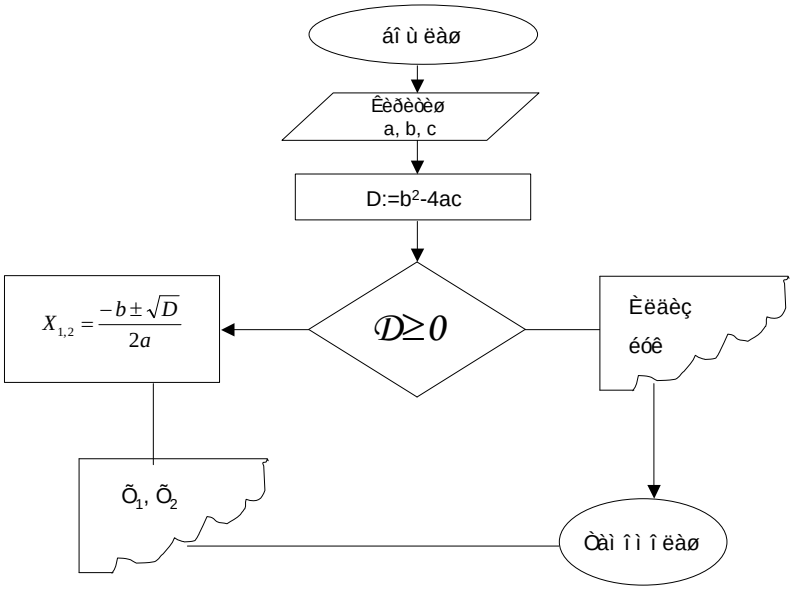
Romb, shart tekshirishni belgilaydi va shart bajarilsa "ha", tarmoq bo’yicha, aks holda "yo’q”-tarmog’i bo’yicha amallar bajarilishini ta’minlaydi.

- Strelka - amallar ketma ketligining bajarilish yo’nalishini ko’rsatadi.

Blok-sxemalar bilan ishlashni yaxshilab o’zlashtirib olish zarur, chunki bu usul algoritmlarni ifodalashning qulay vositalaridan biri bo’lib programma tuzishni osonlashtiradi, programmalash 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 2.1 punktda keltirilgan $ax^2+bx+c=0$ kvadrat tenglamani echish algoritmining blok-sxemasi quyida keltirilgan.

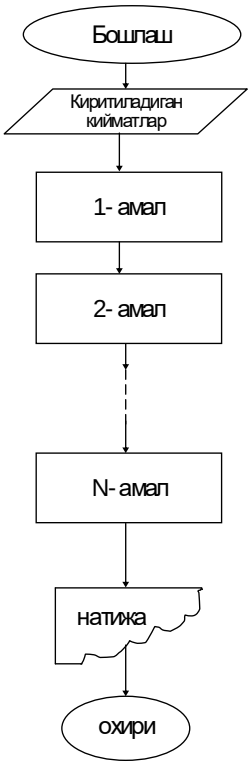


Chiziqli algoritmlar

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 tsiklik algoritmlar,
- ichma-ich joylashgan tsiklik algoritmlar,
- rekurrent algoritmlar,
- takrorlanishlar soni oldindan no’malum algoritmlar,
- ketma-ket yaqinlashuvchi algoritmlar.

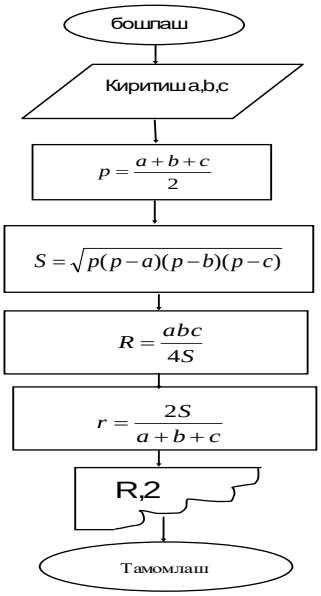
Faqat ketma-ket bajariladigan amallardan tashkil topgan algoritmlarga-chiziqli algoritmlar deyiladi. Bunday algoritmni ifodalash uchun ketma-ketlik strukturasi ishlatiladi. Strukturada bajariladigan amal mos keluvchi shakl bilan ko’rsatiladi. CHiziqli algoritmlarning blok - sxemasini umumiy strukturasini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



1-misol. Uchburchak tomonlarining uzunligi bilan berilgan. Uchburchakka ichki va tashqi chizilgan aylanalar radiuslari va uzunliklari hisoblansin.

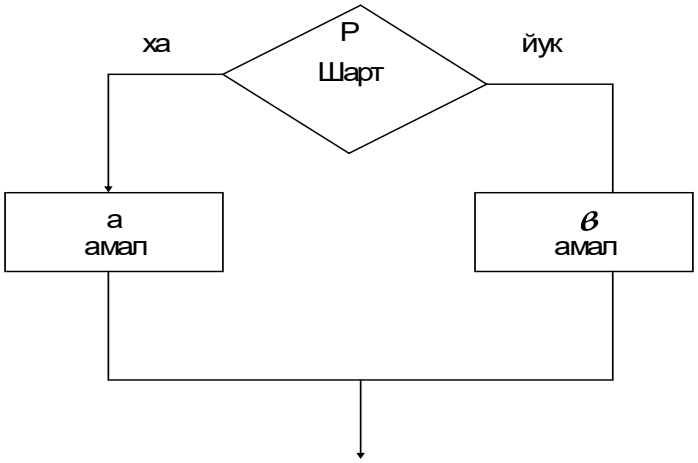
Ichki chizilgan aylana radiusi r = 2S/(a+b+c) tashqi chizilgan aylananing radiusi R = $\frac{4S}{abc}$ formulalar orqali hisoblanadi. Bu erda S uchburchakning yuzi, a, b, c-uchburchak tomonlarining uzunliklari.

Blok-sxemani tuzamiz.



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.

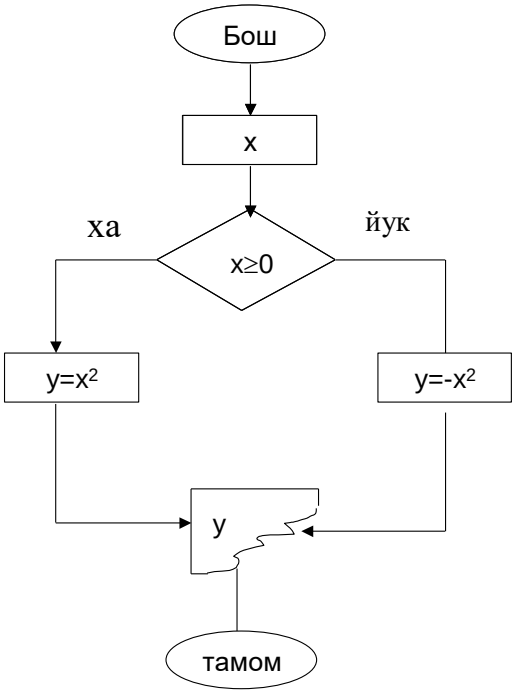


Berilgan shart romb orqali ifodalanadi, r-berilgan shart. Agar shart bajarilsa, "ha" tarmoq bo'yicha a amal, shart bajarilmasa "yo'q" tarmoq bo'yicha b amal bajariladi.

Tarmoqlanuvchi algoritmga tipik misol sifatida quyidagi sodda misolni qaraylik.

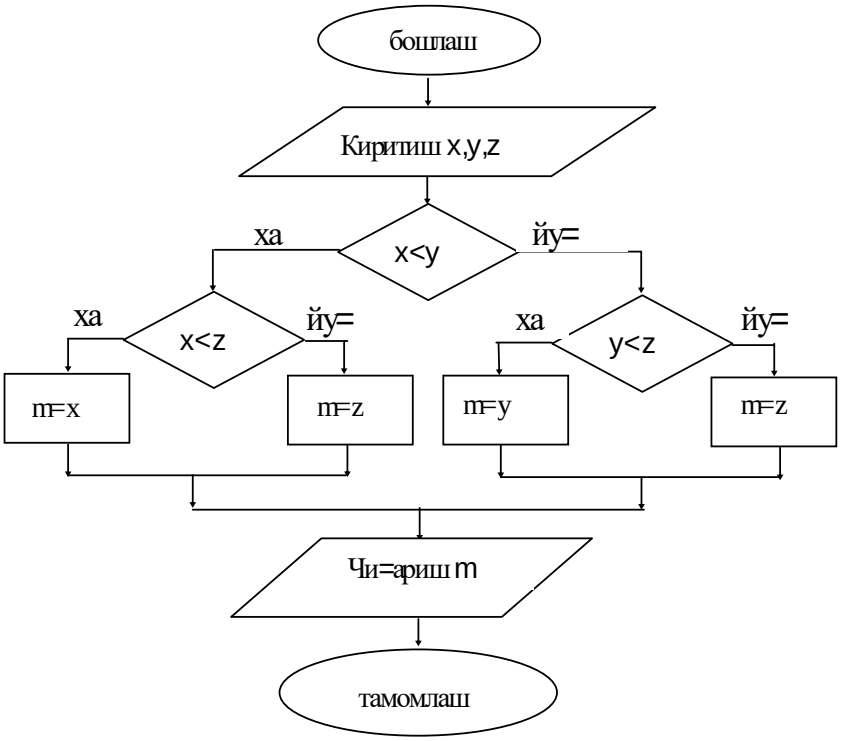
1- Misol**.**

$$Y = \begin{cases} x^2 & \text{agar} \quad x \geq 0 \\ -x^2 & \text{agar} \quad x < 0 \end{cases}$$

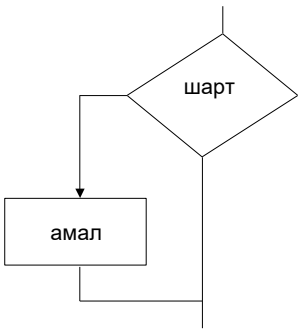


Berilgan x ning qiytmatiga bog'lik holda, agar u musbat bo'lsa «ha» tarmoq bo'yicha $u=x^2$ funktsiyaning qiymati, aks holda $u=-x^2$ funktsiyaning qiymati hisoblanadi.

2-misol. Berilgan x, y, z sonlari ichidan eng kichigi aniqlansin. Berilgan x, y, z sonlardan eng kichigini m-deb belgilaylik. Agar x<u bo'lib, x<z shart bajarilsa, m=x bo'ladi, aksincha x>z shart bajarilsa, m= z bo'ladi. Agar x>u bo'lib, u<z shart bajarilsa, m=u bo'ladi, aksincha u>z shart bajarilsa, m=z bo'ladi. Bu fikrlar quyidagi blok - sxemada o'z aksini topgan. Bu blok–sxemada tarmoqlanish yoki ayri strukturasidan 3 marta foydalanilgan.



Ko’pgina masalalarni echishda, shart asosida tarmoqlanuvchi algoritmlarning ikkita tarmog’idan bittasining ya’ni yoki «ha» eki «yo’q» ning bajarilishi etarli bo’ladi. Bu holat tarmoqlanuvchi algoritmning xususiy holi sifatida aylanish strukturasi deb atash mumkin. Aylanish strukturasi qo’yidagi ko’rinishga ega:



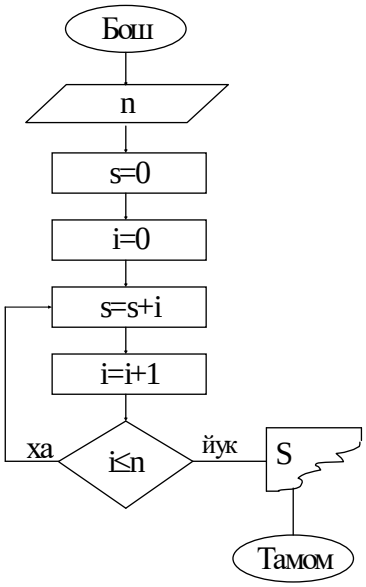
Takrorlanuvchi algoritmlar

Agar biror masalani echish uchun tuzilgan zarur bo’lgan amallar ketma-ketligining ma’lum bir qismi biror parametrga bog’lik ko’p marta qayta bajarilsa, bunday algoritm takrorlanuvchi algoritm yoki tsiklik algoritmlar deyiladi. Takrorlanuvchi algoritmlarga tipik misol sifatida odatda qatorlarning yig’indisi yoki ko’patmasini hisoblash jarayonlarini qarash mumkin. Quyidagi yig’indini hisoblash algoritmini tuzaylik.

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

Bu yig’indini hisoblash uchun i=0 da S=0 deb olamiz va i=i+1 da $S=S+i^2$ ni hisoblaymiz. Bu erda birinchi va ikkinchi qadamlar uchun yig’indi hisoblandi va keyingi qadamda i parametr yana bittaga orttiriladi 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.

1. N –berilgan bo’lsin,
2. i=0 berilsin,
3. S=0 berilsin,
4. i=i+1 hisoblansin,
5. S=S+i hisoblansin,
6. i<N tekshirilsin va bu shart bajarilsa, 4-satrga qaytilsin, aks holda keyingi qatorga o’tilsin,
7. S ning qiymati chop etilsin.



Yuqorida keltirilgan algoritm va blok sxemadan ko’rinib turibdiki amallar ketma-ketligining ma’lum qismi parametr i ga nisbatan N marta takrorlanyapti.

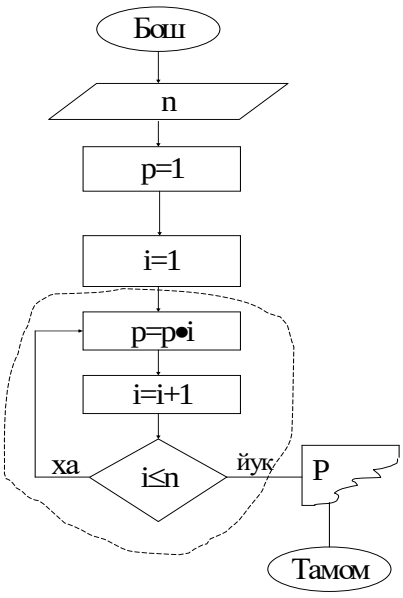
Endi quyidagi ko’paytmaning algoritmini va blok sxemasini tuzib ko’raylik.(1 dan N bo’lgan sonlarning ko’paytmasini odatda P! kabi belgilanadi va faktorial deb ataladi)

$$P = 1 \cdot 2 \cdot 3 \cdots N = P!$$

P! - faktorialni quyidagi ko’rinishda ham yozish mumkin $P = \prod_{i=1}^{N} i$

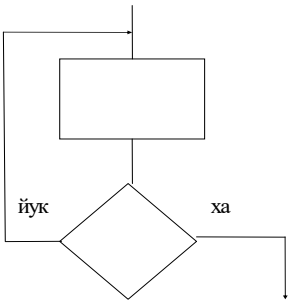
Ko’paytmani hosil qilish algoritmi ham yig’indini hosil qilish algoritmiga juda o’xshash, faqat ko’paytmani hosil qilish uchun i=1 da P=1 deb olamiz va keyin i=i+1 da $P=P*i$ ni hisoblaymiz. Keyingi qadamda i parametr yana bittaga orttiriladi va navbatdagi raqam avvalgi hosil bo’lgan ko’paytma P ga ko’paytiriladi va bu jarayon shu tartibda to I<N sharti bajarilmaguncha davom ettiriladi va natijada izlangan ko’paytmaga ega bo’lamiz. Quyidagi algoritmda bu fikrlar o’z aksini topgan.

1. N–berilgan bo’lsin,
2. i=1 berilsin,
3. P=1 berilsin,
4. i=i+1 hisoblansin,
5. P=P*i hisoblansin,
6. I<N tekshirilsin va bu shart bajarilsa, 4-satrga qaytilsin, aks holda keyingi qatorga o’tilsin,
7. P ning qiymati chop etilsin.

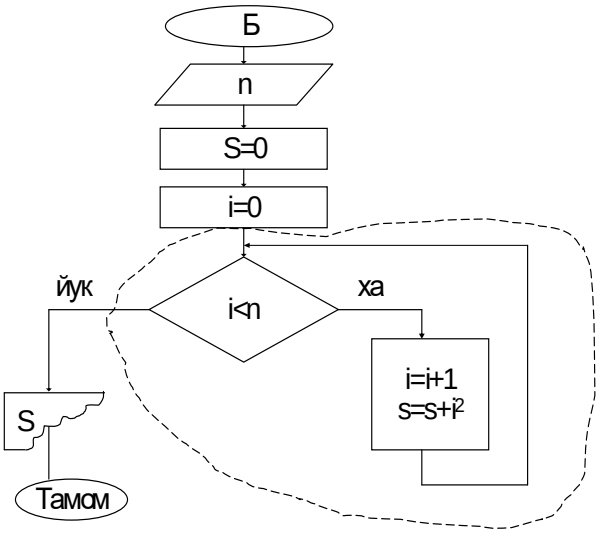


Yuqorida ko’rilgan yig’indi va

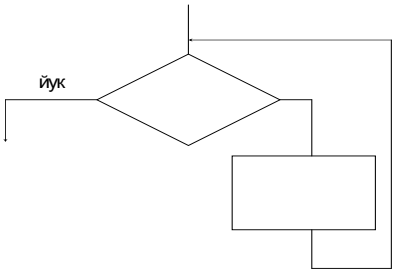
ko’paytmalarning blok sxemalaridagi takrorlanuvchi qismlariga (aylana ichiga olingan) quyidagi sharti keyin berilgan tsiklik struktura mos kelishini ko’rish mumkin.



Yuqoridagi blok sxemalarda shartni oldin tekshiriladigan holdatda chizish mumkin edi. Masalan, yig’indining algoritmini qaraylik.

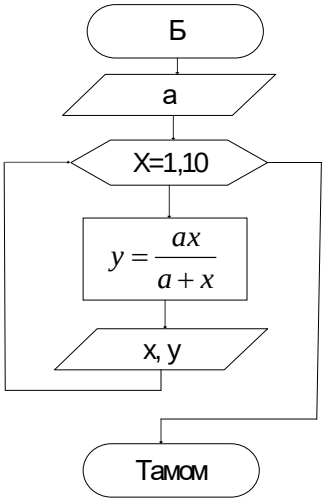


Bu blok sxemaning takrorlanuvchi qismiga quyidagi, sharti oldin berilgan tsiklik strukturaning mos qilishini ko’rish mumkin.



Blok sxemalarining takrorlanuvchi qismlarini, quyidagi parametrik tsiklik strukturasi ko’rinishida ham ifodalash mumkin

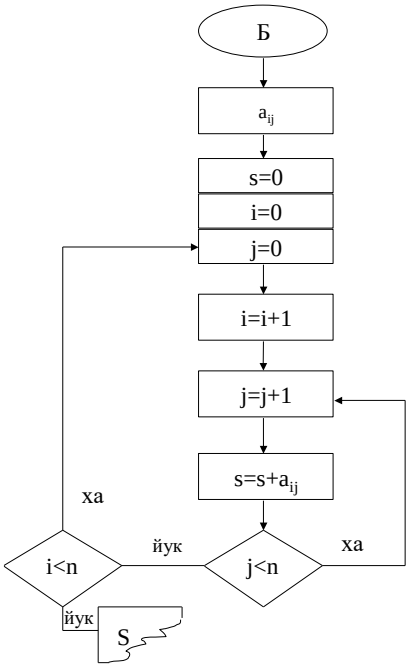
Parametrik tsikl strukturasiga misol sifatida berilgan x=1,2,3,.....10 larda $y = \frac{ax}{a+x}$ funktsiyasining qiymatlarini hisoblash blok sxemasini qarash mumkin.



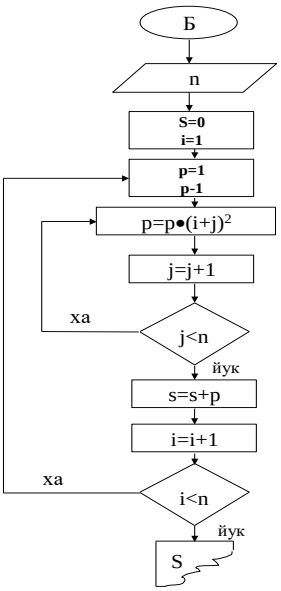
Ichma-ich joylashgan tsiklik algoritmlar

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.

1-misol. $S = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}$ Bu erda i- matritsaning satri nomeri, j-esa ustun nomerini ifodalaydi. YUqoridagi yig’indi ifodagiga mos ravishda, satr elementlari yig’indisini ketma-ket hisoblash zarur bo’ladi. YUqoridagi blok-sxemada shu algoritm ifodalangan.

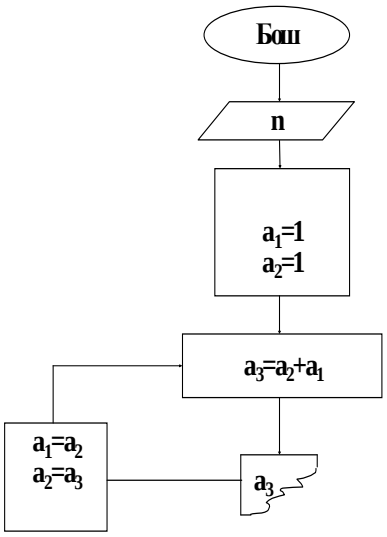


2 misol. $S = \sum_{i=1}^{n} \prod_{j=1}^{n} (i+j)^2$ Bu yig’indi hisoblash uchun, i ning har bir qiymatida j bo’yicha ko’paytmani hisoblab, avval yig’indi ustiga ketma-ket qo’shib borish kerak bo’ladi. Bu jarayon quyidagi blok–sxemada aks ettirilgan. Bu erda i-tashqi tsikl - yig’indi uchun, j-esa ichki tsikl-ko’paytmani hosil qilish uchun foydalanilgan.



Rekurrent algoritmlar.

Hisoblash jarayonida ba’zi bir algoritmlarning o’ziga qayta murojaat qilishga to’g’ri keladi. O’ziga–o’zi murojaat qiladigan algoritmlarga rekkurent algoritmlar yoki rekursiya deb ataladi.



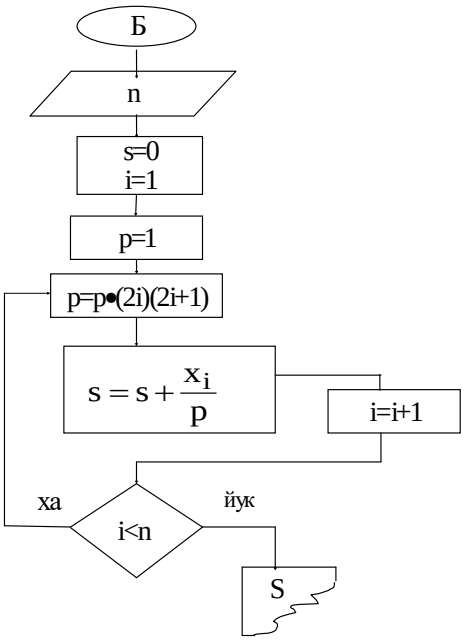
Bunday algoritmga misol sifatida Fibonachchi sonlarini keltirish mumkin. Ma’lumki, Fibonachchi sonlari quyidagicha aniqlangan.

1-misol. $a_0=a_1=1$, $a_i=a_{i-1}+a_{i-2}$ i= 2,3,4,

Bu rekkurent ifoda algoritmiga mos keluvchi blok-sxema yuqorida keltirilgan. Eslatib, o’tamiz formuladagi i-indeksga hojat yo’q, agar Fibonachchi sonining nomerini ham aniqlash zarur bo’lsa, birorta parametr-kalit kiritish kerak bo’ladi.

2-misol. $S=\sum_{i=1}^{n}\frac{x_i}{(2i+i)!}$

Bu ifoda i ning har bir qiymatida faktorialni va yig’indini hisoblashni taqozo etadi. SHuning uchun avval faktorialni hisoblashni alohida ko’rib chiqamiz. Quyidagi rekkurent ifoda faktorialni kam amal sarflab qulay usulda hisoblash imkonini beradi.



R=1

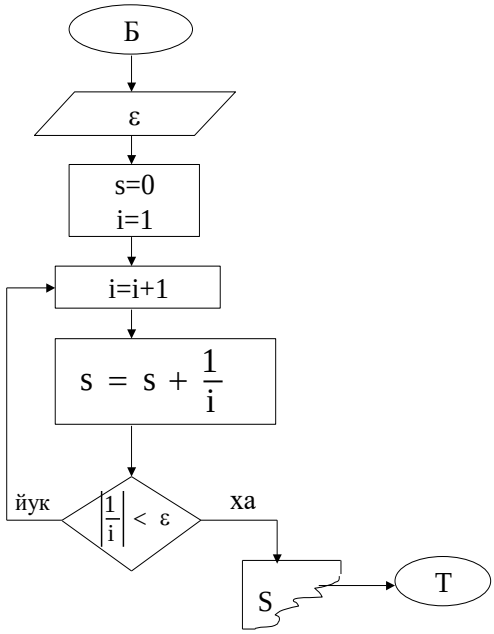
R=R*2i*(2i+1)

Haqiqatan ham, i=1 da 3! ni, i=2 da R=3!*4*5=5! ni va hakozo tarzda (2i+1)! ni yuqoridagi rekkurent formula yordamida hisoblash mumkin bo’ladi. Bu misolga mos keluvchi blok-sxema quyida keltirilgan.

Takrorlanishlar soni no’malum bo’lgan algoritmlar.

Amalda shunday bir masalalar uchraydiki, ularda takrorlanishlar soni oldindan berilmagan-noma’lum bo’ladi. Ammo, bu jarayonni tugatish uchun biror bir shart berilgan bo’ladi.

Masalan, quyidagi $S = 1 + \frac{1}{2} + \frac{1}{3} + ... = \sum_{i=1}^{\infty} \frac{1}{i}$ qatorda nechta had bilan chegaralanish berilmagan. Lekin qatorni ε aniqlikda hisoblash zarur bo’ladi. Buning uchun $\left| \frac{1}{i} \right| < \varepsilon$ shartni olish mumkin.



Ketma-ket yaqinlashuvchi yoki iteratsion algoritmlar.

Yuqori tartibli algebraik va transtsendent tenglamalarni echish ususllari yoki algoritmlari ketma-ket yaqinlashuvchi – interatsion algoritmlarga misollar bo’la oladi. Ma’lumki, transtsendent tenglamalarni echishning quyidagi asosiy usullari mavjud:

- Urinmalar usuli (Nyuton usuli),
- Ketma-ket yaqinlashishi usuli,
- Vatarlar usuli,
- Teng ikkiga bo’lish usuli.

Bizga f(x)=0 (1) transtsendent tenglama berilgan bo’lsin. Faraz qilaylik bu tenglama [a,b] oraliqda uzluksiz va f(a) f(b)<0 shartni qanoatlantirsin. Ma’lumki, bu holda berilgan tenglama [a,b] orilaqda kamida bitta ildizga ega bo’ladi va u quyidagi formula orqali topiladi.

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)} \qquad n = 0,1,2,..... \quad (2)$$

Boshlang’ich X_0 qiymat $f(x_0)f''(x_0) < 0$ shart asosida tanlab olinsa, (2) iteratsion albatta yaqinlashadi. Ketma-ketlik

$$\left| X_{n+1} - X_n \right| < \varepsilon$$

shart bajarilgunga davom ettiriladi.

1-Misol. Berilgan musbat a xaqiqiy sondan kvadrat ildiz chiqarish algoritmi tuzilsin.

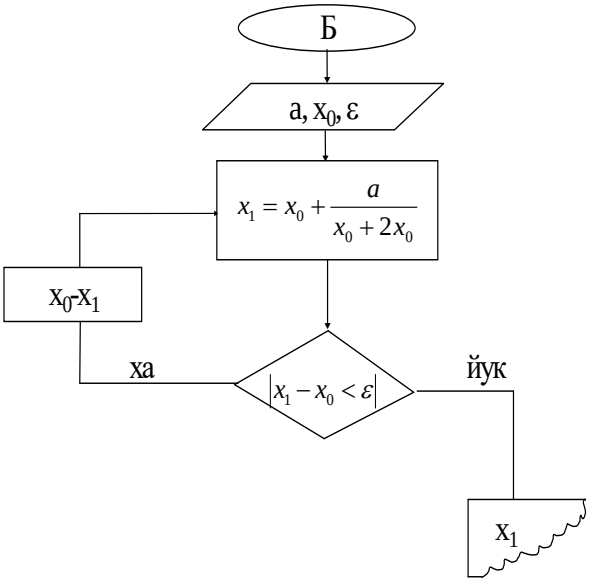
Bu masalani echish uchun kvadrat ildizni x deb belgilab olib, $\sqrt{a} = x(3)$ ifodalash yozib olamiz. U holda (1) tenglamaga asosan

$$f(x) = x^2 - a \qquad (4)$$

Ekanligini topish mumkin (4) ifodani (2) ga qo'yib, quyidagi rekurrent formulani topish mumkin.

$$X_{n+1} = \frac{1}{2}(X_n + \frac{a}{2X_n}) \qquad (5)$$

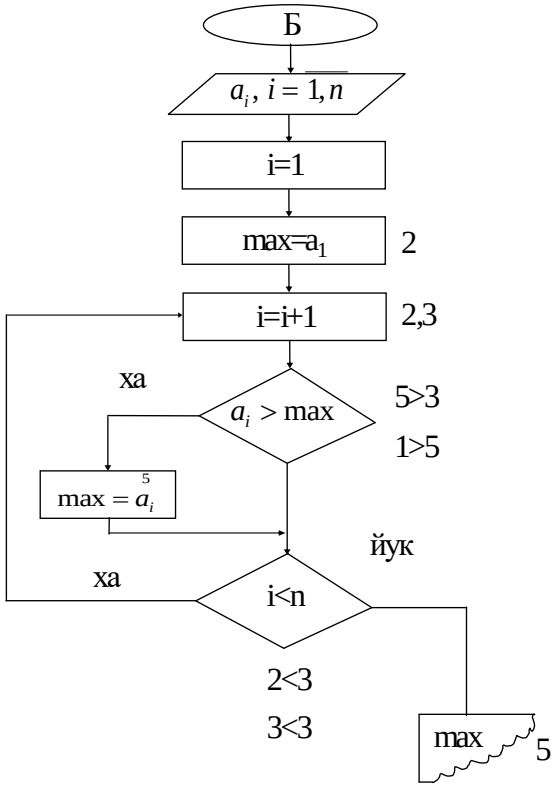
Bu formulaga mos blok-sxema quyida keltirilgan. ε - kvadrat ildizni topishning berilgan aniqligi. Eslatib o'tamiz, algoritmda indeksli o'zgaruvchilarga zarurat yo'q.



Algoritm ijrosini tekshirish.

Kompyuter uchun tuzilgan algoritm ijrochisi-bu kompyuterdir. Biror programmalash tilida yozilgan algoritm kodlashtirilgan oddiy ko'rsatmalar ketma-ketliligа o'tadi va mashina tomonidan avtomatik ravishda bajariladi. Metodik nuqtai–nazardan qaraganda algoritmning birinchi ijrochisi sifatida o'quvchining o'zini olish muhim ahamiyatga ega. O'quvchi tomonidan biror masalani echish algoritmi tuzilganda bu algoritmni to'g'ri natija berishini tekshiri juda muhimdir. Buning yagona usuli o'quvchi tomonidan algoritmni turli boshlang'ich berilganlarda qadamma - qadam bajarib (ijro etib) ko'rishdir. Algoritmni bajarish natijasida xatolar aniqlanadi va to'g'rilanadi. Ikkinchi tomonidan, masalani echishga qiynalayotgan o'quvchi uchun tayyor algoritmni bajarish – masalani echish yo'llarini tushunishga xizmat qiladi.

Algoritm ijrosini quyidagi misolda ko'raylik.Berilgan a_i, $i=1,n$ sonlarning eng kattasini topish algoritmini tuzaylik. Buning uchun, berilgan sonlardan birinchisi a_1 ni $i=1$ eng katta qiymat deb faraz qilaylik va uni max nomli yangi o'zgaruvchiga uzataylik: max=a_1. Parametr i ning qiymatini bittaga oshirib, ya'ni i=i+1 a_1 ni a_2 bilan taqqoslaymiz va qaysi biri katta bo'lsa uni max o'zgaruvchisiga uzatamiz va jarayon shu tarzda to i=n bo'lguncha davom ettiramiz. Bu fiklar quydagi blok-sxemada o'z aksini topgan.



Endi bu blok-sxema yoki algoritmning ijrosini $n=3$ $a_1=3, a_2=5,$ $a_3=1$ Aniq sonlarda qadamma–qadam ko'rib o'taylik:

1. i=1 da max=3 bo'ladi.
2. i=i+1=2 ni topamiz,
3. a_2>max, ya'ni 5>3 ni tekshiramiz, shart bajarilsa, max=5 bo'ladi.
4. i<n 1 ni 2<3 ni tekshiramiz. SHart bajarilsa, i ni yana bittaga oshiramiz, va i=3 bo'ladi, va
5. a_3 max, ya'ni 1>5, ni tekshiramiz. SHart bajarilmadi, demak, keyingi
6. i<n shartni, ya'ni 3<3 ni tekshiramiz. SHart bajarilmadi. Demak max=5 chop etiladi. Biz blok-sxemani tahlil qilish davomida uning to'g'riligicha ishonch hosil qildik. Endi ixtiyoriy n lar uchun bu blok-sxema bo'yicha eng katta elementni topish mumkin.

XULOSA

Xulosa qilib aytganda Algaritm bilan ishlashish barcha turdagi dasturlash tillarida ishlash imkoniyatini yengillashtirib beradi. Har bir dasturning dastlab algaritmini yaratib olgan maqul. Agar biz dasturimizning ketma ketligini bilmasak, u dastur biz oylagandan koproq hajmni egallashi mumkin ekan. Men C++ dasturi strukturasi haqida, belgilar bayoni, algoritm va dastur tushunchasi, ma'lumotlarni kiritish va chiqarish operatorlari hamda dasturda ishlatiladigan toifalar, ifodalar va ko'nikmalarga ega bo`ldim. Algoritmlash va dasturlash tillari bo'yicha yozilgan bir necha kitoblar bilan tanishib chiqdim va ulardan o'zimga kerakli malumotlarni oldim. Kurs ishimda programmalash texnologiyalari masalalari, algoritmlar, ularning xossalari, tasvirlash usullari va tipik algoritmlarga blok sxemalar tuzish masalalari qaralgan.

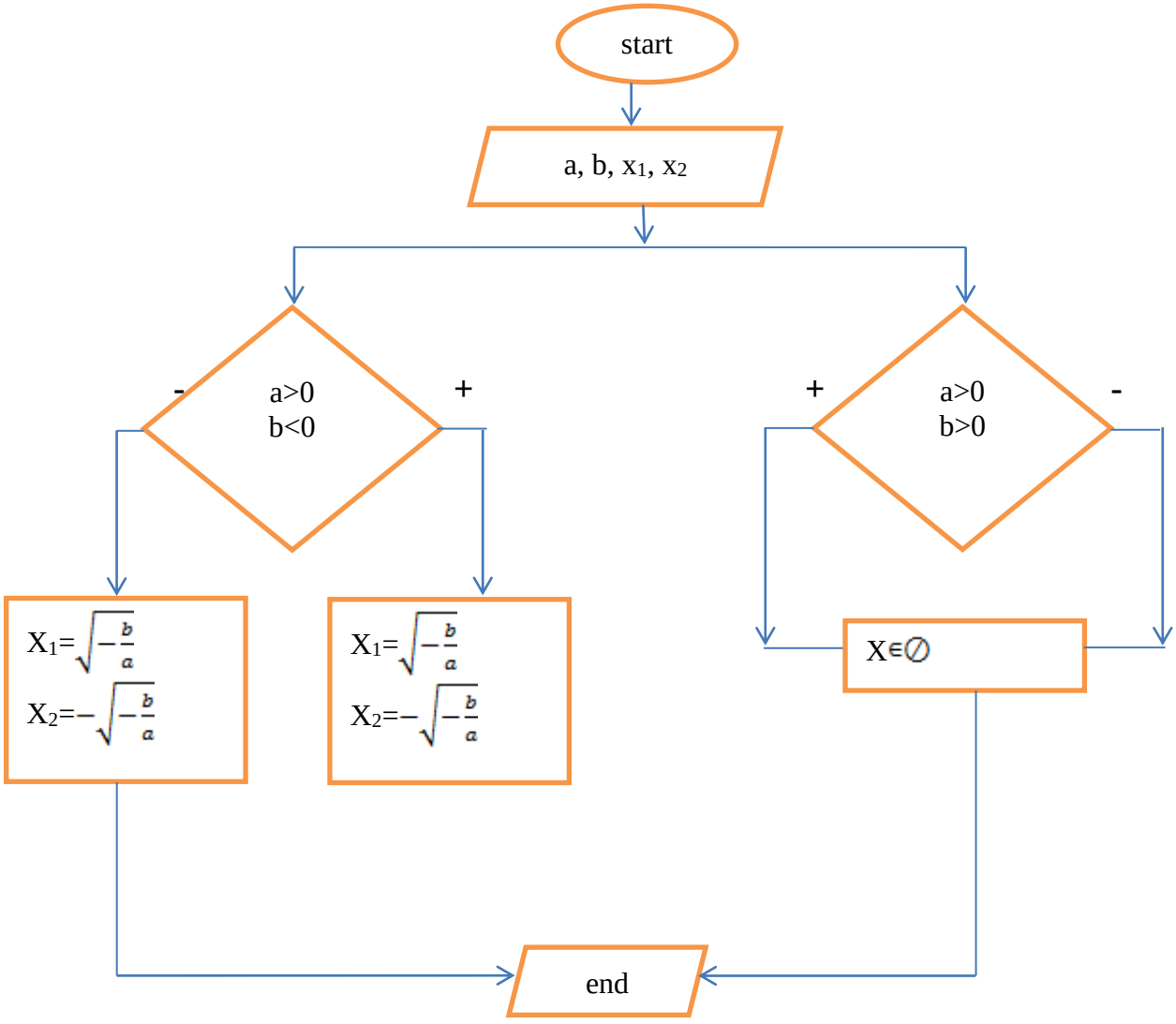
Amaliy bo’lim.

1-Amaliy ish.

Masalaning qoyilishi:

$ax^2+b=0$ Tenglamaning yechimi topilsin.

Blok sxemasi:



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

 int main()
 {
     float a,b,x1,x2;
     cin>>a;
     cin>>b;
     if (a>0 and b<0)
     {
         x1=sqrt(-b/a);
         x2=-sqrt(-b/a);
     cout<<"x1="<<x1<<endl;
     cout<<"x2="<<x2<<endl;
     }
     if (a<0 and b>0)
     {
     x1=sqrt(-b/a);
     x2=-sqrt(-b/a);
         cout<<"x1="<<x1<<endl;
     cout<<"x2="<<x2<<endl;
     }
      if (a>0 and b>0 )
      {
          cout<<"Yechimga ega emas"<<endl;
      }
      if (a<0 and b<0 )
      {
         cout<<"Yechimga ega emas"<<endl;
      }
     return 0;
 }
```

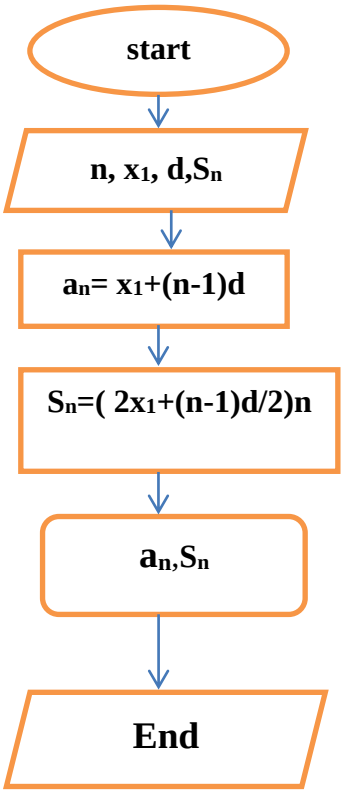
```
25
-9
x1=0.6
x2=-0.6

Process returned 0 (0x0)   execution time : 3.721 s
Press any key to continue.
```

2- Amaliy ish

Masalaning qoyilishi:

1. **n ta natural son berilgan. 1- azosi x_1 va ayirmasi d bolgan arifmetik progressiyaning nta xadi yig'indisini toppish.**

Blok sxemasi:



Masalaning ko'rinishi.

main.cpp

```cpp
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int n, x1, d, S, y;
    cout<<"x1="; cin>>x1;
    cout<<"d="; cin>>d;
    cout<<"n="; cin>>n;
    y=x1+(n-1)*d;
    S=(2*x1+(n-1)*d)*n/2;
    cout<<"y="<<y<<endl;
    cout<<"S="<<S<<endl;
    return 0;
}
```

```
x1=2
d=3
n=10
y=29
S=155

Process returned 0 (0x0)   execution time : 10.608 s
Press any key to continue.
```

3-Amaliy ish.

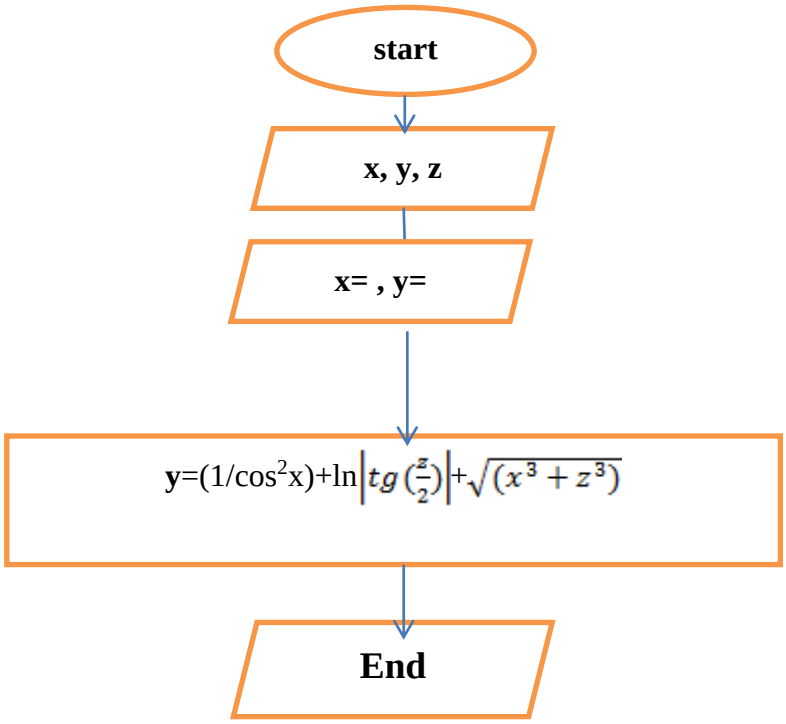
Masalaning berilgani:

$Y=(1/\cos^2 x)+\ln\left|tg\left(\frac{z}{2}\right)\right|+\sqrt{(x^3+z^3)}$

Masalaning algoritmi:

Matematik amallar va formulalardan foydalangan holda ynii topamiz.

Blok sxemasi:



Masalaning ko'rinishi

Y=(1/cos²x)+ln$\left|tg\left(\frac{z}{2}\right)\right|+\sqrt{(x^3+z^3)}$

main.cpp

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    float x, y, z;
    cout<<"x="; cin>>x;
    cout<<"z="; cin>>z;
    y=1/(cos(x)*cos(x))+log(fabs(tan(z/3)))+sqrt(x*x*x+z*z*z);
    cout<<"y="<<y<<endl;
    return 0;
}
```

```
x=3
z=2
y=6.69667

Process returned 0 (0x0)   execution time : 1.529 s
Press any key to continue.
```

FOYDALANGAN ADABIYOTLAR.

1. Markushevich A. I. Teoriya analiticheskix funktsiy. V 2-x t. – M.: Nauka, 1968. T.2. – 624s
2. Goluzin G.M. Geometricheskaya teoriya funktsii kompleksnogo peremennogo. – M. : Nauka, 1976.– 540 s.
3. B. V. SHabat. Vvedenie v kompleksnıy analiz. 1–chast. M.N. 1989.
4. G. Xudayberganov, A. Vorisov, X. Mansurov. Kompleks analiz. Toshkent, «Universitet», 1998.
5. G. Xudayberganov, A. Vorisov, X. Mansurov. Kompleks analiz.Karshi. «Nasaf», 2003.
6. Virt N. Algoritmı strukturı dannıx programmı.-M.:Mir, 1985.-405s.
7. Aripov M.M., Imomov T., Irmuxamedov Z.M. va boshqalar. Informatika. Axborot texnologiyalari. Toshkent, 1-qism. 2002, 2-qism. 2003
8. http://ziyonet.uz
9. www.google.uz