

A.R. Azamatov

А36 об/аз
408209



ALGORITMLASH VA DASTURLASH ASOSLARI



O'ZBEKISTON RESPUBLIKASI OLIY VA O'RTA
MAXSUS TA'LIM VAZIRLIGI
O'RTA MAXSUS, KASB-HUNAR TA'LIMI MARKAZI

A.R. AZAMATOV

ALGORITMLASH VA DASTURLASH ASOSLARI

Kasb-hunar kollejlari uchun o'quv qo'llanma

Ikkinchi nashri

*Cho'lpon nomidagi nashriyot-matbaa ijodiy uyi
Toshkent — 2011*

UDK: 519.711(075)

BBK 22.12ya722

A36

*Oliy va o'rtta maxsus kasb-hunar ta'limi o'quv metodik
birlashmalar faoliyatini muvofiqlashtiruvchi Kengash
nashrga tavsiya etgan*

Taqrizchi:

B. Boltayev – fizika-matematika fanlari nomzodi.

Azamatov, A. R.

A36

Algoritmash va dasturlash asoslari: kasb-hunar kollejlari uchun o'quv qo'llanma/A.R. Azamatov; O'zbekiston Respublikasi Oliy va o'rtta maxsus ta'lim vazirligi, O'rtta maxsus, kasb-hunar ta'limi markazi. Ikkinchi nashri. – T.: Cho'lpon nomidagi nashriyot-matbaa ijodiy uyi, 2011. – 232 b.

ISBN 978-9943-05-439-4

O'quv qo'llanma «Algoritmash va dasturlash asoslari» o'quv fani dasturi asosida yozildi. Unda algoritmash asoslarini o'rgatish uchun Ijrochilar va ularning ko'rsatmalar sistemasi kiritilgan bo'lib, algoritm tushunchasi, algoritm xossalari, ularni tasvirlash va tuzish usullari haqida ma'lumotlar yoritilgan. Bundan tashqari, ko'rsatmalarni dasturlash tillarida ifodalari izohlanib, BASIC, PASCAL va DELPHI dasturlash tillarida taqqoslamalar berilgan. Nazariy ma'lumotlardan tashqari takrorlash va mustaqil ishlash uchun ko'p sonli vazifalar keltirilgan.

Ushbu o'quv qo'llanma kasb-hunar kollejlari va akademik litsey talabalari uchun mo'ljallangan bo'lib, undan algoritmash va dasturlashni o'rganmoqchi bo'lgan barcha kitobxonlar ham foydalanishlari mumkin.

UDK: 519.711(075)

BBK 22.12ya722

ISBN 978-9943-05-439-4

© Cho'lpon nomidagi nashriyot-matbaa ijodiy uyi, 2010

© Cho'lpon nomidagi nashriyot-matbaa ijodiy uyi, 2011

I bob. ALGORITMLAR

Algoritm tushunchasi

Inson hayoti davomida katta-kichik vazifalar yoki masalalarni hal etishni o'z oldiga maqsad qilib qo'yadi. Odatda, u o'z maqsadiga erishishi uchun bajarishi lozim bo'lgan amal yoki ishlarini hayotiy tajribasi yoki o'zlashtirgan bilimiga asoslanib ma'lum bir tartibga keltiradi. Bunga hayotimizdan xilma-xil misollar keltirish mumkin.

1.1-misol

Ko'chadan o'tish maqsad qilib qo'yilgan bo'lsin. U holda ko'chadan o'tayotgan kishi hammamizga odatiy hol bo'lib qolgan quyidagi harakatlarni bajarishi lozim bo'ladi:

- 1) chap tarafga qaralsin, agar transport vositasi yo'q bo'lsa, 2-bandga o'tilsin, aks holda 1-bandga o'tilsin;
- 2) o'ng tarafga qaralsin, agar transport vositasi yo'q bo'lsa, 3-bandga o'tilsin, aks holda 1-bandga o'tilsin;
- 3) ko'chadan o'tilsin.

1.2-misol

Eni 6 metr va bo'yi 10 metr bo'lgan joyni to'ldirish uchun sotib olinishi kerak bo'lgan 12x25 sm (eni 12 sm va bo'yi 25 sm) g'ishtlar soni topilsin.

Hisoblayotgan kishi geometriya fanidan olgan bilimiga asoslanib quyidagi ketma-ketlikdagi amallarni bajaradi:

- 1) joyning yuzasi S_{joy} santimetr o'lchov birligida topilsin;
- 2) bir dona g'ishtning yuzasi $S_{g'isht}$ santimetr o'lchov birligida topilsin;
- 3) g'ishtlar soni S_{son} joyning yuzasini g'ishtning yuzasiga nisbati deb olinsin.

Bu amallar ketma-ketligini quyidagi matematik formula bilan ham ifodalash mumkin:

$$S_{son} = \frac{S_{joy}}{S_{g'isht}} = \frac{6 \cdot 100 \cdot 10 \cdot 100}{12 \cdot 25}.$$

1.3-misol

Amal bajarisin: 19632107 + 19702202. Bu amalni qanday bajargan bo'lar edingiz? Ha, to'g'ri, bu sonlarni ustun ko'rinishida deyarli quyidagicha qo'shasiz:

1) sonlar xonalari to'g'ri keladigan tartibda birinchisining tagiga ikkinchisi yozib olinsin;

2) sonlarning birlik xonasidagi raqamlarini qo'shib, natijani birlik xonasidagi raqami birliklar tagiga yozilib, o'nlik xonasi raqami dilda saqlansin;

3) sonlarning o'nliklardagi va dildagi raqamlarni qo'shib, natijani birlikdagi raqami o'nliklar tagiga yozilib, o'nlik raqami dilda saqlansin;

va 3-banddagi qoida yuzliklar, mingliklar va hokazo uchun takrorlanadi. Bu amallar quyidagi korinishda sizga juda tanish:

$$\begin{array}{r} 19632107 \\ + 19702202 \\ \hline 39334309 \end{array}$$

Yuqoridagi misollarda keltirilgan amallar ketma-ketligi, boshqacha aytganda, ko'rsatmalar yoki buyruqlar ketma-ketligi biror kishi tomonidan bajarilgach, ko'zlangan maqsadga erishiladi. Bunday amallar ketma-ketligi yoki hayotimizda har kuni va har soatda uchrab turadigan turli qoidalar ichida biror zaruriy natijaga erishishga olib keladigan amallarni ketma-ket bajarishni talab etadigan qoidalar informatikaning asosiy tushunchalaridan biri **algoritm** so'zi bilan ifodalanadi.

Algoritm so'zi IX asrda yashab (783- yilda tug'ilgan) o'z ilmiy ishlari xazinasini bilan dunyoga tanilgan vatandoshimiz buyuk astronom, matematik va geograf Abu Abdullo Muhammad ibn Muso al-Xorazmiy nomidan kelib chiqqan. Al-Xorazmiy arifmetikaga bag'ishlangan «Hind hisobi haqida kitob» risolasida to'q-qizta hind raqamining sonlarni ifodalashdagi afzalliklari va ular yordamida har qanday sonni ham qisqa va oson yozish mumkinligini aytadi va hozirgi kunda hamma o'quvchilar biladigan sonlar ustida, yuqoridagi 3-misoldagi kabi ustun ko'rinishida amallar bajarish qoidalarini yoritadi. Ayniqsa, nol (0) qo'llashning ahamiyati haqida tushuncha berib, nolni yozmaslik natijaning xato chiqishiga olib keladi, degan. Bu risola XII asrda Ispaniyada lotin tiliga tarjima qilingan va butun Yevropaga tarqatilgan. Bu tarjimaning XIV asrda ko'chirilgan qo'lyozmasining yagona nusxasi Kembrij universitetining kutubxonasida saqlanmoqda. Risola

«*Dixit Alxhorithmi*», ya'ni «*Dediki al-Xorazmiy*» iborasi bilan boshlanadi.



Algoritm deganda, biror maqsadga erishishga qaratilgan iirochi bajarishi uchun mo'liallangan ko'rsatma (bu^vrua)larning aniq, tushunarli va chekli ketma-ketligi tushuniladi.

Bu algoritm tushunchasining matematik ta'rifi bo'lmasa ham intuitiv ma'noda algoritmning mazmunini ochib beruvchi tavsifidir. Algoritmni intuitiv ma'noda bir necha misollarda izohlaymiz. Biror-bir narsani taqiqlovchi qoidalar algoritm bo'lolmaydi, masalan: «Chekish mumkin emas», «Begonalarning kirishi taqiqlanadi», «Kirish», «Chekish uchun joy» kabi biror-bir narsaga ruxsat etuvchi qoidalar ham algoritmgga xos emas. Lekin «Svetoforni yashil rangida o'ting» juda sodda bo'lsa ham algoritmdir.

Demak, yuqorida keltirilgan misollardagi ko'rsatmalar ketma-ketligi **algoritm** va bu algoritmlarni bajarayotgan inson – **ijrochi** bo'lar ekan. Algoritm ijrochisi faqat insonmi, degan savol berishingiz tabiiy. Bu savolga javob quyidagicha:



Algoritm ijrochisi – algoritmga ko'rsatilgan bu^vrua voki ko'rsatmalarni bajara oladigan abstrakt voki real (texnik voki biologik) sistema.

Ijrochi bajara olishi uchun algoritm unga tushunarli bo'lishi lozim. Algoritm ijrochi tushunadigan tilgagina emas, balki uning bilim va malakasiga ham mos bo'lishi kerak. Aks holda ijrochi birorta ham ko'rsatmani bajara olmasligi mumkin.

Ijrochi bajara olishi mumkin bo'lgan ko'rsatma yoki buyruqlar to'plami **ijrochining ko'rsatmalar sistemasini** deyiladi. Masalan, «16 sonidan kvadrat ildiz chiqarilsin» ko'rsatmasi 2-sinf o'quvchisining ko'rsatmalar sistemasiga tegishli bo'lmaydi, lekin 8-sinf o'quvchisining ko'rsatmalar sistemasiga tegishli bo'ladi. Algoritm ijrochiga tushunarli bo'lishi uchun ijrochining imkoniyatlarini bilish lozim. Agar ijrochi inson bo'lsa, u holda algoritm insonning imkoniyatlaridan kelib chiqib tuzilishi kerak. Bunda ko'zlangan maqsad va algoritmdan kelib chiqib inson tushunadigan til, insonning bilimi, hayotiy tajribasi, kasbiy malakasi, yoshi, qolaversa, jismoniy imkoniyatlari hisobga olinishi zarur. Agar ijrochi texnik vosita (masalan, kompyuter, elektron soat, dastgohlar) bo'lsa, u holda algoritm shu texnik vositaning imkoniyatlaridan kelib chiqib tuzilishi kerak.

Agar ijrochi kompyuter hisoblanib, uning dasturiy ta'minotida berilgan («Karra jadvalini hosil qilish») algoritmni bajara oladigan

dastur (masalan, elektron jadvallardan birortasi ham) bo'lmasa, u holda hech qanday natijaga erishib bo'lmaydi.

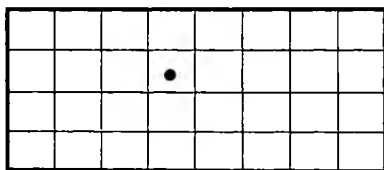
Demak, berilayotgan har qanday ko'rsatma ijrochining ko'rsatmalar sistemasidan olinishi, ya'ni ijrochi uni qanday bajarishni bilishi kerak ekan. Bu algoritmnning **tushunarlilik** xossasini ifodalaydi.

Shuni ta'kidlash joizki, informatikada algoritmnning asosiy ijrochisi sifatida kompyuter xizmat qiladi.

Ijrochi

Bu qo'llanmada algoritm tuzish usullarini o'rgatish uchun sizni bir nechta Ijrochi bilan tanishtiramiz [1], lekin ular kompyuter yoki inson emas, balki biz uchun abstrakt sistema. Misol sifatida **Robot** nomli ijrochi bilan tanishtiramiz.

Robot teng o'lchamdagi kvadratlarga bo'lingan tekislikda yashaydi (1.1-rasm). U kvadratlarning birida joylashgan va ixtiyoriy qo'shni kvadratga o'tishi mumkin. Shu bilan birga Robot o'zi turgan kvadratni bo'yashi mumkin.



1.1-rasm.

Robot 5 ta ko'rsatmani bajaradi:

yuqoriga

quyiga

chapga

o'ngga

bo'ya

Bulardan **yuqoriga**, **quyiga**, **chapga** va **o'ngga** ko'rsatmalari Robotni mos yo'nalishlar bilan siljishga majbur qiladi. Lekin **bo'ya** ko'rsatmasida Robot harakatlanmaydi, faqat o'zi turgan kvadratni bo'yaydi. Agar kvadrat bo'yalgan bo'lsa u holda **bo'ya** ko'rsatmasida kvadratning rangi o'zgarmaydi.

Robotning hayotidagi voqealardan eng qizig'i, ba'zi kvadratlar orasida devor borligi (1.2-rasm). Odatda, Robot har tomondan devorlar bilan o'ralgan va kvadratlardan hosil bo'lgan to'g'ri

to'rtburchak ichida joylashgan bo'ladi. Lekin shu to'g'ri to'rtburchak ichida ham devorlar bo'lishi mumkin.

Ba'zan devorlar murakkab shaklni hosil qiladi, bu shaklni **labirint** deb atashadi.

Robot devor ichidan o'ta olmaydi. Agar devor ichidan o'tmoqchi bo'lsa, u holda Robot «sochilib» ketadi.



1.2-rasm.

Bunday halokatli holatlarga tushmaslik uchun quyidagi to'rtta shartni tekshirish zarur:

yuqori bo'sh

quyi bo'sh

chap bo'sh

o'ng bo'sh

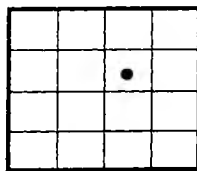
Bo'sh so'zi shu tomonda devor yo'qligini bildiradi.

Robot o'zi turgan katakning devorinigina aniqlay oladi. O'zi turgan kvadrat bilan devor orasida bitta kvadrat bo'lsa ham uzoqdagi bu devorni «ko'ra» olmaydi. U yonida turgan devorgagina «tegib» ko'rishi mumkin. 1.3-rasmda turli holatlarda yuqori bo'sh degan birgina shartning qiymatini ko'rish mumkin. Tushunarliki, yuqori bo'sh sharti (yoki yuqori bo'sh da'vosi Rost bo'lsa) Robot yuqoriga ko'rsatmasini «sochilib» ketmasdan bajara olishini bildiradi.

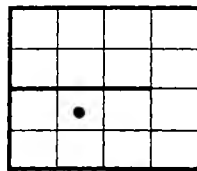
yuqori bo'sh



YOLG'ON



ROST



YOLG'ON

1.3-rasm.

Bu kabi mulohazalar chap bo'sh sharti va chapga ko'rsatmasi, yana boshqa juftliklar uchun ham to'g'ri.

Ro'yxatni yakunlash uchun Robot biladigan oxirgi shartni keltiramiz:

bo'yalgan

Bu shart Robot turgan kvadratni bo'yalgan yoki bo'yal-maganligini tekshirish imkonini beradi. Agar kvadrat bo'yalgan bo'lsa, shart ROST, aks holda YOLG'ON bo'ladi.

Ko'rib turibsiz, Robotning ko'rsatmalari juda sodda. Lekin uni o'rab turgan muhit xilma-xil imkoniyatlarga boy. Robotning maydonida turli labirintlar, yo'laklar, har xil shakldagi xonalar va boshqa figuralar yordamida juda ko'p qiziqarli masalalar qo'ysih mumkin. Robotning mikrohayoti – algoritmik tafakkurni rivojlan-tirish uchun a'lo darajadagi mashq maydonidir.

Ijrochilarni boshqalari bilan tanishtirishdan avval ularni nimalar farqlab turishini izohlab o'tmoqchimiz.

Ijrochini quyidagilar farqlab turadi:

- ijrochi muhiti;
- ijrochining ko'rsatmalar sistemasi;
- sodda amallar;
- INKOR.

Ijrochi muhiti – ijrochi «yashaydigan» yoki algoritmnii baja-radigan muhiti. Ijrochi Robot misolida bu katakli maydon, bo'yalgan kataklar va devorlar. Ularning joylashishi va Robotning turgan joyi muhitning aniq holatini beradi.

Har bir ijrochi qat'iy belgilangan ro'yxatdagi – ijrochinining ko'rsatmalar sistemasidagi – ko'rsatmalarni bajara oladi. Har bir ko'rsatma uchun qo'llash sharti (muhitning qanday holatida ko'rsatmani bajarish mumkinligi) va ko'rsatmani bajarilish natijasi belgilangan bo'lishi kerak. Masalan, **yuqoriga** ko'rsatmasi Ro-botning yuqorisida devor yo'q bo'lsagina bajarish mumkin. Bu ko'rsatmani bajarilish natijasi – Robot yuqoriga bitta katak siljiydi.

Ko'rsatma chaqirilgandan keyin Ijrochi **sodda amal** bajaradi. Robot misolida – yuqoriga bitta katak siljish.

INKOR – bu holat bo'lib, ko'rsatma muhitning mumkin bo'lmagan holatida chaqirilganda yuz beradi. Robot misolida qarasa, agar u devor ichidan o'tmoqchi bo'lsa, «sochilib» ketadi va bu Robot uchun INKOR holatiga olib keladi.

Yodingizda bo'lsin: Ijrochi algoritm maqsadi haqida hech narsa bilmaydi, u berilgan ko'rsatmalarni so'zsiz bajaradi, xolos.

Algoritmning xossalari

Algoritmning tavsifida «biror maqsadga erishishga qaratilgan» jumlasini qo'llanilgan. Bu maqsadni yuqorida keltirilgan misollarda ko'rishimiz mumkin: ko'chadan o'tish, g'ishtlar sonini hisoblash, yig'indini hisoblash. Bular algoritmning natijaviylik (cheklilik) xossasi bilan bog'liq. Bu xossaning mazmuni shundan iboratki, har qanday algoritm ijrochi chekli qadamdan so'ng oxir-oqibat ma'lum bir yechimga olib kelishi kerak. Shuni ta'kidlash joizki, algoritm avvaldan ko'zlangan maqsadga erishishga olib kelmasligi ham mumkin. Bunga ba'zan algoritmning noto'g'ri tuzilgani yoki boshqa xatolik sabab bo'lishi mumkin. Ikkinchi tomondan, qo'yilgan masala ijobiy yechimga ega bo'lmasligi ham mumkin. Lekin salbiy natija ham natija deb qabul qilinadi.

1.4-misol

$x^2+x+1=0$ kvadrat tenglama yechilsin.

Bu tenglamaga quyida keltirilgan « $ax^2+bx+c=0$ ($a \neq 0$) ko'rinishidagi kvadrat tenglamani yechish» algoritmini qo'llab, tenglama yechimga ega emasligini aniqlaymiz. Bu ham *natija* ekanligi sizga ma'lum.

1) diskriminant: $D = b^2 - 4ac$ hisoblansin;

2) agar $D < 0$ bo'lsa, tenglama yechimga ega emas deb olinsin va 5-bandga o'tilsin;

3) agar $D = 0$ bo'lsa, yagona yechim $-\frac{b}{2a}$ ga teng deb olinsin va 5-bandga o'tilsin;

4) birinchi yechim $\frac{-b - \sqrt{D}}{2a}$ ga, ikkinchi yechim $\frac{-b + \sqrt{D}}{2a}$ ga teng deb olinsin;

5) tugallansin.

1.1-masha

E'tibor oilgan bo'lsangiz, diskriminantning noldan kichikligi, nolga tengligi tekshirildi, ammo noldan kattaligi tekshirilmadi. Sababini o'vlab ko'ring!

Demak, algoritm doimo chekli qadamdan iborat bo'lishi va biror natija berishi kerak ekan. Bu algoritmnini **diskretlilik** (uzluksizlik),

alohidalik) xossasiga olib keladi. Algoritmida masalani yechish jarayoni alohida olingan sodda ko'rsatmalar ketma-ketligini qadamba-qadam bajarishdan iborat bo'lishi kerak. Bu xossa misollardan yaqqol ko'rinish turibdi. «Angliyada avtomobilni yo'lning chap qismida haydang» qoidasi talabnoma bo'lgani bilan uzluksizlik xarakteriga ega va shuning uchun ham algoritm hisobiga qo'shilmaydi.

E'tiboringizni yana bir narsaga qaratamiz. Keltirilgan misollarda quyidagi jumlar bor: «Ko'chadan o'tish» (ariqdan yoki dengizdan emas), «Eni 6 metr va bo'yi 10 metr bo'lgan joyni» (kilometr emas), «eni 12 santimetr va bo'yi 25 santimetrli g'ishtlar» (eni 5 santimetr va bo'yi 100 santimetrli g'ishtlar emas), « $x^2+x+1=0$ kvadrat tenglama» ($x^2-1=0$ kvadrat tenglama emas). Bu jumlar va qavs ichida yozilganlarni taqqoslasangiz, olinadigan natija shu jumalardagi «qiymat»larga chambarchas bog'liq ekanligini tushunasiz. Agar bu «qiymatlar» o'zgarsa, masalan, qavs ichidagilarga, olingadigan natija umuman boshqacha bo'lishini ko'rish qiyin emas. Qiymat so'zini qo'shtirnoq ichiga olganimizga sabab, siz doimo qiymat so'zini faqat sonlar bilan bog'lab o'rganib kelgansiz. Lekin, bilingki, biror masala uchun qiymat har xil turdagi obyektlar bo'lishi mumkin ekan. Demak, har bir algoritmning natijasi avvaldan berilgan, ya'ni boshlang'ich qiymatlarga bog'liq bo'lar ekan. Boshlang'ich qiymatlar turli natijalarga olib kelishiga yana bir hayotiy misolga o'zingiz javob bera olasiz: sizga va mehmonga kelgan do'stlaringizga pishiralayotgan palovga 20 gramm tuz solish o'rniga 200 gramm tuz solishsa natija bir xil bo'ladimi?

Har bir algoritm — bu amallarni belgilovchi qoida bo'lib, ularning zanjiri natijasida biz boshlang'ich qiymatlardan izlangan natijaga kelamiz. Bunday amallar zanjiri algoritmik jarayon, har bir amal — algoritmning qadami deb ataladi.

Yana boshlang'ich qiymat va natija bo'lishi mumkin bo'lgan narsalarning tahliliga qaytamiz. Ko'rdikki, har bir algoritm uchun boshlang'ich qiymatlarning turli hollarini tanlash mumkin. Masalan, g'ishtlar masalasi algoritmi uchun boshlang'ich qiymatlarni tavsiflashda «santimetr» so'zlarini «uzunlik o'lchovlari» kabi tushunish mumkin. Bu holda hosil bo'ladigan natijaning miqdori o'zgaradi, xolos. Ko'p algoritmlar boshlang'ich qiymatlarning turli hollari uchun o'z kuchini saqlab qoladi. Qo'shish algoritmini ixtiyoriy natural sonlar jufti uchun qo'llash mumkin. Avval aytib o'tilgan algoritmlarni aniqlangan bu xossasi (ularni boshlang'ich qiymatlar-

ning juda ko'p sondagi hollariga qo'llash mumkinligi) **ommaviylik** deb ataladi. Yuqorida keltirilgan « $ax^2+bx+c=0$ ($a\neq 0$) ko'rinishidagi kvadrat tenglamani yechish» algoritmi ixtiyoriy a , b , c haqiqiy sonlar uchun natija beradi, ya'ni algoritmning ommaviylik xossasi o'rinli ekan. Algoritmni juda ko'p xususiy hollarda ishlashi shunday o'ta muhim va ahamiyatli xossa, shu sababli ancha vaqtgacha uni algoritmning ilmiy ta'rifiga kiritilishi shart deb hisoblendi. Bu ko'pgina qoidalarni fan sohasidan chiqarib tashlar (ularni «oz miqdordagi» ommaviyligi tufayli) va ham ilmiy tadqiqotni, ham ularning natijasini amaliyotda qo'llashni qiyinlashtirar (balki bu ilmiy bo'lmagan hol bo'lsachi?), bu esa jiddiy noqulaylikka sabab bo'ladi.

Boshlang'ich qiymatlarning faqat bitta – yagona holiga qo'llash mumkin bo'lgan algoritm ham katta ahamiyat kasb etadi. Ularga turli xil avtomatlardan (masalan, agar aniq bir tangaga moslangan gazeta sotadigan avtomat, yoki telefon-avtomat) foydalanish algoritmni tegishlidir, aniq bir joydan boshlanadigan va belgilangan joyga olib boradigan yo'nalish bo'yicha borish algoritmi va boshqalar.

«Ommaviylik» atamasining mavhumligi mashhur, ba'zan uyum paradoksi deb ataluvchi, Evbulid paradoksi orqali tasdiqlanadi. Paradoks mazmunini o'zimizga savol berib va javobini berib tezda aniqlab olishimiz mumkin. Bitta tosh – uyummi? Yo'q. Ikkita tosh-chi? Yana yo'q. Uchtasi-chi? Oxir-oqibat, biz yoki uyum mavjud emas degan xulosaga, yoki shunday sondagi toshlar to'plami borki, undan bitta oshsa uyum hosil bo'lishiga olib keladi, deb e'tirof etishga majbur bo'lamiz. U yoki bu fikr ham haqiqatga ziddir va bu uyum atamasining mavhumligining natijasi bo'lib hisoblanadi. Nima bo'lganda ham ommaviylik xossasidan oddiygina «bo'yin tovlash» mumkin emas. Uni ta'riflashni ozgina o'zgartirish hamda shuning asosida yuqorida aytib o'tilgan mavhumlikni yo'qotish zarur.

«Ommaviylik» atamasiga qanday mazmun kiritish kerak? Bu savolga javob shunday. Har bir algoritm uchun biror-bir obyektlar (narsalar, buyumlar va hokazo) sinfi mavjud va ularning barchasi boshlang'ich qiymat sifatida olinishi mumkin, deb hisoblash zarur. Manfiymas butun sonlarni qo'shish algoritmi uchun manfiymas butun sonlarning barcha juftligi; avtomatdan «Toshkent oqshomi» gazetasini sotib olish algoritmi uchun – yagona obyekt – tanga shunday sinf bo'la oladi. Algoritmning ommaviyligi – mos sinfning barcha obyektlarini qo'llash mumkinligidir, ya'ni, ularni qandaydir

miqdorining (chekli yoki cheksiz) qo‘llash mumkin emasligi emas. Mumkin bo‘lgan, ya’ni joiz obyektlarni miqdori chekli yoki cheksizligi, yoki miqdori nolga teng bo‘lishi – bu shu sinfning xususiyatidir.

Agar algoritm yordamida joiz boshlang‘ich qiymat asosida izlangan natijani olish mumkin bo‘lsa u holda **algoritmni joiz boshlang‘ich qiymatga qo‘llash mumkin** deyiladi. Agar boshlang‘ich qiymat joiz bo‘lsa ham natija olish mumkin bo‘lmasa, u holda unga **algoritm qo‘llash mumkin emas** deyiladi.

Endi joiz boshlang‘ich qiymatlar sinfi qanday ekanligini ko‘rib chiqamiz. Boshlang‘ich qiymatlar ba‘zan narsa yoki buyumlar, sonlar ekanini ko‘rdik. Bu fikr olingan natijalar uchun ham o‘rinli. Bu narsalar orasidagi umumiylik nimada? Algoritm – bu qoidalar va demakki, ular qandaydir tillarda ifodalangan, degan fikrni e‘tiborga olsak, bu umumiylik ko‘rinadi. Bir necha marta bu qoidalarni aniq bajarilishi qanchalik muhim ekanligi haqida gapirib o‘tdik.

Lekin bunday aniq bajarilishi boshlang‘ich qiymatlar (ular bilan birga izlangan natijalar ham) biror-bir tilda, balki yangisida, batamom tavsiflanishga imkon bersagina mumkin. Bu holda har bir boshlang‘ich qiymatga, har bir oraliq natijaga va nihoyat, izlangan natijaga qandaydir gap mos keladi. Yana, mazkur gapning «Mazmun»i bir qiymatli bo‘lishi zarur.

Matematikada ko‘pincha maxsus usul qo‘llanadi. Bu usul shundan iboratki, biror-bir obyekt boshqa tabiatli obyekt bilan almashtiriladi, bunda yangi obyektlarga birlamchilari bilan bir qiymatli mos bo‘ladi. Ko‘rilayotgan holda boshlang‘ich qiymatlar tilining gaplari bilan boshlang‘ich qiymatlarning o‘zi orasida bir qiymatli moslik mavjud. Shu sababli, algoritmni matematik ta’riflashda boshlang‘ich qiymatlar va izlangan natijalar tilning gaplari deb hisoblanishi mumkin.

Bunday almashtirish amaliyot nuqtayi nazaridan mumkinmi? Albatta, mumkin. Chunki, algoritmning o‘zida boshlang‘ich qiymatlar emas, ularning **nomi**, jarayonni bajarish uchun esa amallar va hosil bo‘ladigan natijalarning **nomini** bilish yetarli.

Keltirilgan usul algoritm ta’rifini tor ma’noda bo‘lishiga olib keladi, deyish mumkin. Bunday fikr asoslidir. Lekin bu torayish muhim emas, chunki u algoritmlar beradigan imkoniyatlarni kamaytira olmaydi.

Bu kabi yondashish boshlang‘ich qiymatlar va natijalar turlarini nisbatan kamaytiradi, ammo ular avvalgidek turli fizik

tabiatga ega bo'lishi mumkin, lekin biz uchun bu, ularni nazariy qaraganimizda, turli tillardagi gaplar kabidir. Narsalarning turlanishini biz tillarning turlanishiga keltirdik. To'g'ri, tillar ham kam emas. Ularni cheksiz to'plam (faqat mavjudlari emas, balki mavjud bo'lishi mumkin bo'lganlari ham, ya'ni mumkinlari ham) deb hisoblash mumkin. Lekin har bir algoritm faqat ikkita til bilan bog'langan: bittasida u ta'riflangan, ikkinchisining gaplari boshlang'ich qiymatlar hollarini uning uchun mumkin bo'lganlaridir. Birinchi tilni, odatda, **algoritmik til** deb, ikkinchisini – **operandlar tili** deb atashadi. **Operandlar** deb shunday obyektlarga aytiladiki, ular ustida algoritm talab qilgan amallar bajariladi. Operandlar tilining barcha gaplari joiz deb hisoblanadi, bu tilga tegishli bo'lmagan biror-bir belgilar birikmasi ta'rif bo'yicha joiz emas.

Algoritmni tasvirlash usullari

Algoritmnlarni tasvirlashning turli usullaru mavjud. Quyida algoritmnlarni tasvirlashning keng tarqalgan usullarini ko'rib chiqamiz.

1. Algoritmning so'zlar yordamida ifodalanishi

Avval keltirilgan bir qator misollar inson og'zaki nutqida qo'llaniladigan so'zlar orqali ifodalangan edi (masalan, ko'chadan o'tish algoritmi, g'ishtlar sonini hisoblash algoritmi). Algoritmning bunday tasvirlash usulida ijrochi uchun ko'rsatma jumlarlar orqali ko'rsatma shaklida beriladi. Qo'llanmada, asosan, shu usuldan foydalanamiz.

2. Algoritmning formulalar yordamida ifodalanishi

Bu usul matematika, fizika, kimyo va biologiya kabi fanlarda ko'plab qo'llaniladi. Yodingizda bo'lsa, so'zlar yordamida ifodalangan g'ishtlar sonini hisoblash algoritmini formula orqali ifodalagan edik. Formuladagi «+», «-», «x», «:» kabi arifmetik amallarning tartibiga rioya qilgan holda bajarilishi ham algoritmga misol bo'ladi. Avval berilgan « $ax^2 + bx + c = 0$ ($a \neq 0$) ko'rinishidagi kvadrat tenglamani yechish» algoritmining quyidagi formula orqali ifodasi bilan tanishsiz:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Bilasiz, formuladagi amallar ma'lum bir tartib bilan bajarilishi shart.

3. Algoritmning jadval yordamida ifodalanishi








Algoritmning bu ko'rinishda berilishi ham sizga tanish. Masalan, matematikada qo'llanib kelinayotgan Bradis jadvali deb nomlangan to'rt honali matematik jadval, lotareya yutuqlar jadvali, Mendeleyev kimyoviy elementlar jadvali. Bunday jadvallardan foydalanish ma'lum bir algoritm qo'llashni talab etadi.

Biror funksiyaning grafigini chizish uchun ham funksiyaning argument qiymatlariga mos qiymatlar jadvalini hosil qilamiz. Bu ham algoritmning jadval ko'rinishiga misol bo'ladi.

4. Algoritmning grafik shaklda ifodalanishi

Algoritmning bu ko'rinishda ifodalanishi matematikada chizilgan grafik, kerakli uyni oson topish uchun dahalarda o'rnatilgan uylarning joylashish sxemasi, avtobuslarning yo'nalish sxemasi orqali sizga tanish.

Algoritmni asoslarini o'rganishning yana bir qulay grafik shakli – **blok-sxema** usulidir. Blok-sxemalar bir yoki bir nechta buyruq yoki ko'rsatmani aks ettiruvchi maxsus geometrik shakllar – bloklardan tashkil topadi. Bloklar yo'nalish chiziqlari orqali tutashiriladi.

	algoritmning boshlanishini va tugallanganligini bildiradi
	ma'lumotlarni kiritish yoki chiqarishni bildiradi
	oddiy harakatni, ya'ni qiymat berish yoki tegishli ko'rsatmalar berishni bildiradi
	shart tekshirilishini bildiradi
	yordamchi algoritmgga murojaatni bildiradi
	blok-sxemadagi harakat yo'nalishini bildiradi
	qiymat berish ko'rsatmasi

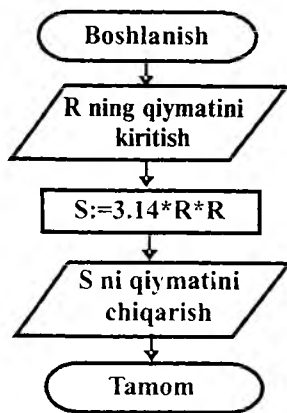
1.5-misol

R radiusli doiraning yuzasini hisoblash algoritmi tuzilsin.

Avval aytib o'tilganidek, algoritmda boshlang'ich qiymatlar o'rniga ularning nomlari ishtirok etishi mumkin va bu algoritmning ommaviylik xossasiga aloqadorligini bildiradi. Bu masalada ham radiuslar guruhi R nomi bilan berilmoqda va uning joiz boshlang'ich qiymati ixtiyoriy haqiqiy son bo'lishi mumkin. Eslatib o'tamiz, algoritmda turli nomlar ishtirok etishi va ular boshlang'ich qiymatlar va natijalar nomi bo'lishi ham mumkin. Masalaning quyida keltirilgan yechimidagi S nomi masalani yechimi bo'ladigan natijalar guruhining nomidir.

Bu masala algoritmini ikki xil usulda: so'zlar yordamida va grafik shaklda tuzamiz:

- 1) Boshlanish;
- 2) R ning qiymati aniqlansin;
- 3) R ning R ga ko'paytirib, S deb olinsin;
- 4) S ni 3,14 ga ko'paytirib, S deb olinsin;
- 5) javob sifatida S yozilsin;
- 6) tugallansin.



5. Algoritmning dastur shaklida ifodalanishi

Ma'lumki, kompyuter dasturlar asosida ishlaydi va boshqariladi. Siz hozirgacha MS Word, MS Paint va MS Excel kabi amaliy **dasturlar** bilan ishladingiz. Lekin har bir amaliy **dastur** ham juda katta va murakkab algoritmning bir ko'rinishidir. Demak, bu kabi algoritmlar bajarilishi uchun ular **algoritm ijrochisiga, ya'ni kompyuterga tushunarli bo'lishi lozim**.

Odatda, algoritmning kompyuter tushunadigan tilda yozilishi **dastur** deb ataladi. Kompyuter tushunadigan til esa **dasturlash tili** deb ataladi. Jahonda hozirgi kunda minglab dasturlash tillari mavjud va yana rivojlanib bormoqda. Hozirgi kunda keng tarqalgan va o'rganish uchun qulay bo'lgan BASIC, Pascal, VBA, Delphi, C dasturlash tillari o'rganiladi.

Algoritmning asos turlari

Qo'llanmada, asosan, algoritmuk tafakkurning rivojlantirishini maqsad qilib qo'ygan bo'lsak-da, algoritm to'g'risida tasavvuringizni kengaytirish maqsadida yana ba'zi ma'lumotlarni berishni lozim topdik.

Har qanday algoritm mantiqiy tuzilishiga, ya'ni bajarilishiga qarab uch asosiy turga bo'linadi: **chiziqli (ketma-ketlik)**, **tarmoqlanuvchi** va **takrorlanuvchi**. Algoritmikada bu algoritmlar asosida turli-tuman yangi algoritmlar hosil qilinadiki, ular ham o'z navbatida mustaqil ahamiyatga ega bo'ladi.

Chiziqli algoritmlar. Bu turdagi algoritmlarda hech qanday shart tekshirilmaydi. Shu sababli barcha ko'rsatmalar ketma-ket bajarib boriladi. «G'ishtlar sonini hisoblash», «Doira yuzini hisoblash» algoritmlari chiziqli algoritmlarga misol bo'ladi. Lekin hayotimizdagi juda ko'p jarayonlar shartlar asosida boshqariladi.

Tarmoqlanuvchi algoritmlar. Shartga muvofiq bajariladigan ko'rsatmalar ishtirok etgan algoritmlar **tarmoqlanuvchi algoritmlar** deb ataladi. Algoritmning bu turi hayotimizda har kuni va har qadamda uchraydi. Eshikdan chiqishimiz eshik ochiq yoki yopiqligiga, ovqatlanishimiz qornimiz och yoki to'qligiga yoki taomning turiga, ko'chaga kiyinib chiqishimiz ob-havoga, biror joyga borish uchun transport vositasini tanlashimiz to'lash imkoniyatimiz bo'lgan pulga bog'liqdir. Demak, tarmoqlanuvchi algoritmlar chiziqli algoritmlardan tanlash imkoniyati bilan farqlanar ekan.

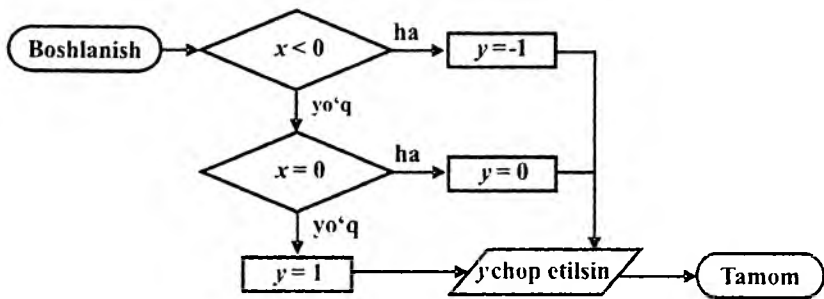
Avval yoritilgan «Ko'chadan o'tish», «Kvadrat tenglamani yechish» algoritmlari ham tarmoqlanuvchi algoritmlarga misol bo'ladi.

1.6-misol

Algoritmi formula yordamida berilgan

$$y = \begin{cases} -1, & \text{agar } x < 0 \\ 0, & \text{agar } x = 0 \\ 1, & \text{agar } x > 0 \end{cases}$$

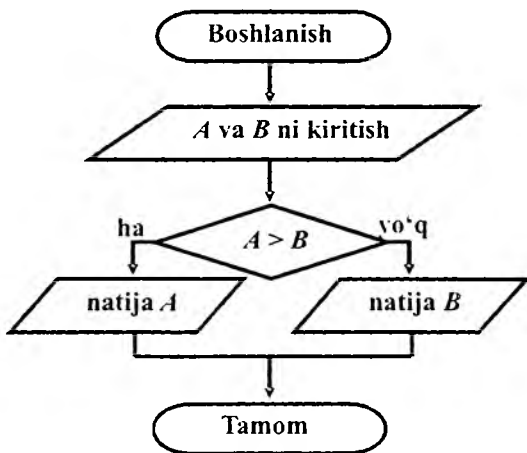
funksiyaning qiymatini hisoblashga doir tarmoqlanuvchi algoritmi blok-sxema yordamida tasvirlaymiz:



1.7-misol

Berilgan ikkita A va B sonlardan kattasini topish (IKT nomi bilan ataluvchi) algoritmini so'zlar va blok-sxema yordamida tuzing.

- 1) Boshlanish;
- 2) A va B kiritilsin;
- 3) agar $A > B$ bo'lsa
- 4-bandga o'tilsin;
- aks holda
- 5-bandga o'tilsin;
- 4) natija A deb olinsin va
- 6-bandga o'tilsin;
- 5) natija B deb olinsin;
- 6) tugallansin.



Bu misoldan quyidagicha xulosa chiqarish mumkin: agar $A > B$ shart bajarilsa 5-banddagi ko'rsatma bajarilmaydi, aks holda, ya'ni $A \leq B$ bo'lsa, 4-banddagi ko'rsatma bajarilmaydi. IKT algoritmi tarmoqlanishni yaqqol tasavvur qilish imkoniyatini beradi.

Takrorlanuvchi (siklik) algoritmlar. Masalalarni tahlil etish jarayonida algoritmdagi ba'zi ko'rsatmalar takroran bajarilishini kuzatish mumkin. Hayotimizda ham juda ko'p jarayonlar takrorlanadi. Masalan, darslarning har hafta takrorlanishi, har kuni nonushta qilish yoki maktabga borish va hokazo. Ko'rsatmalari takroriy bajariladigan algoritmlar **takrorlanuvchi algoritmlar** deb ataladi.

Takrorlanuvchi algoritmlar « $I := I + 1$ », « $S := S + I$ » yoki « $P := P * I$ » ko'rinishidagi ko'rsatmalarning ishtiroki bilan ajralib

turadi (* – ko‘paytirish amali). Bunday ko‘rsatmalarning mazmunini tushunish uchun takrorlanishning bir nechta qadamini ko‘rib chiqamiz.

Odatda, yig‘indi uchun boshlang‘ich qiymat (inglizchadan SUMM, ya‘ni yig‘indi ma‘noli so‘zning bosh harfi) $S := 0$ va ko‘paytma uchun (inglizchadan PRODUCT, ya‘ni ko‘paytma ma‘noli so‘zning bosh harfi) $P := 1$ deb olinadi, chunki bu qiymatlar, ya‘ni 0 va 1 lar, mos ravishda, yig‘indi va ko‘paytmaning natijasiga ta‘sir etmaydi:

1-qadamda $I := 1$ bo‘lsin:

$$S := S + I = 0 + 1 = 1, P := P * I = 1 * 1 = 1;$$

2-qadam: $I := I + 1 = 1 + 1 = 2$:

$$S := S + I = 1 + 2 = 3, P := P * I = 1 * 2 = 2;$$

3-qadam: $I := I + 1 = 2 + 1 = 3$:

$$S := S + I = 3 + 3 = 6, P := P * I = 2 * 3 = 6;$$

4-qadam: $I := I + 1 = 3 + 1 = 4$:

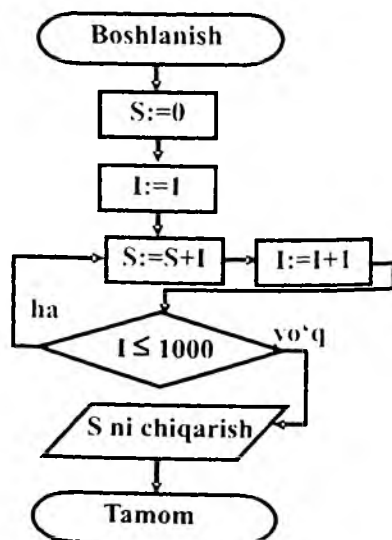
$$S := S + I = 6 + 4 = 10, P := P * I = 6 * 4 = 24.$$

Algoritmikada, matematikada bunday bo‘lishi mumkin emas, $I = I + 1$ deb yozilishi mumkin. Bu yozuvda avval o‘ng tomondagi qiymat hisoblanib, so‘ng bu qiymat chap tomondagi nomning qiymati deb olinadi.

1.8-misol

1 dan 1000 gacha bo‘lgan sonlar yig‘indisini, ya‘ni $S = 1+2+3+...+1000$ ni hisoblash algoritmini tuzing.

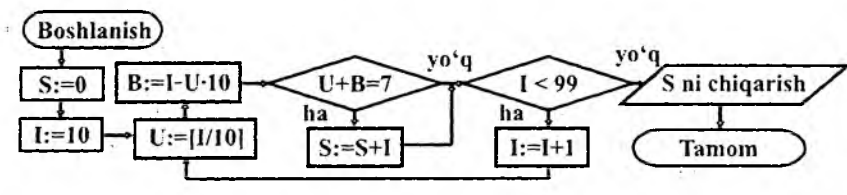
- 1) Boshlansin;
- 2) $S = 0$ deb olinsin
(ya‘ni $S := 0$);
- 3) I ning qiymati 1 deb olinsin
(ya‘ni $I := 1$);
- 4) S ga I qo‘shilib, S deb olinsin
(ya‘ni $S := S + I$);
- 5) I ga 1 qo‘shilib I deb olinsin
(ya‘ni $I := I + 1$);
- 6) agar $I \leq 1000$ bo‘lsa
4-bandga o‘tilsin;
- 7) javob deb S olinsin;
- 8) tugallansin.



So'zlar bilan ifodalangan algoritmda blok-sxema bilan mutanosiblikni bildirish uchun qavslar ichida izohlar berib bordik. Odatda, takrorlanuvchi algoritmlarda « $I := I + 1$ » ifoda **sanagich** deb yuritiladi. Bu misol yechimini chiziqli algoritm shaklida tashkil etish ham mumkin. Buning uchun arifmetik progressiyaning 1000 ta hadi yi'gindisini hisoblash formulasidan foydalanish kifoya (algoritmni mustaqil tuzing). Lekin keyingi misolda bunday qilish ancha mushkul.

1.9-misol

2 xonali sonlar ichidan raqamlari yi'g'indisi 7 ga teng sonlar yi'g'indisini hisoblash algoritmini tuzing ($[a]$ – a sonining butun qismi, $/$ – bo'lish amali).



Ko'rib o'tilgan algoritmlarga nazar tashlasak, algoritmlar chiziqli, tarmoqlanuvchi yoki takrorlanuvchi qismlardan tashkil topganligini kuzatish mumkin. Demak, inson hayotida uchraydigan algoritmlar, asosan, shu uch turdagi algoritmlarning uzviy birligi sifatida namoyon bo'ladi.

Nazorat savollari va topshiriqlar

1. Algoritm nima? Misollar keltiring.
2. Algoritmning qanday xossalari bilasiz?
3. Boshlang'ich qiymatlar deganda nimani tushunasiz?
4. Hamma bajara olishi uchun algoritm qanday xossaga ega bo'lishi kerak?
5. Tushunarlilik xossasi bajariladigan va bajarilmaydigan ko'rsatmalar ketma-ketligiga misollar keltiring.
6. Ko'rsatmalar ijrochiga tushunarli bo'lishi uchun qanday sistemadan olinishi kerak?
7. Ijrochi algoritmni so'zsiz bajarishi uchun qanday xossa ahamiyatli? Javobingizni izohlang.
8. Algoritmning diskretlilik xossasini misollar yordamida tushuntiring.
9. Algoritmning natijaviylik xossasini misollar yordamida tushuntiring.

10. Natijaviylik xossasi bajarilmaydigan ko'rsatmalar ketma-ketligiga misollar keltiring.
11. Algoritmning ommaviylik xossasini misollar yordamida tushuntiring.
12. Algoritmning qanday tasvirlash usullari bor?
13. Algoritmni so'zlar orqali ifoda etilishiga hayotiy misollar keltiring.
14. Qaysi fanlarda algoritmni formulalar yordamida berish qulay?
15. Algoritmni formulalar orqali ifoda etilishiga fizika fanidan misollar keltiring.
16. Algoritmni jadval ko'rinishida berilishiga misollar keltiring.
17. Algoritmni grafik shaklda berilishiga misollar keltiring.
18. Blok-sxemaning asosiy elementlariga misollar keltiring.
19. Algoritmni dastur shaklida berilishiga misollar keltiring.
20. Qanday algoritmlar chiziqli algoritm deb ataladi? Chiziqli algoritmlarga hayotiy misollar keltiring.
21. Qanday algoritmlar tarmoqlanuvchi algoritm deb ataladi? Tarmoqlanuvchi algoritmlarga hayotiy misollar keltiring.
22. Qanday algoritmlar takrorlanuvchi algoritm deb ataladi? Takrorlanuvchi algoritmlarga hayotiy misollar keltiring.
23. Chiziqli, tarmoqlanuvchi va takrorlanuvchi algoritmlarning bir-biridan farqini tushuntiring.
24. Uchta sondan kattasini (UKT) aniqlab beruvchi algoritm tuzing.
25. Berilgan sonning ishorasini aniqlovchi algoritm tuzing.
26. Quyidagi ko'rsatmalar ketma-ketligi qanday algoritm turiga misol bo'ladi? Algoritmlar natijasini aniqlang:
 - a) $a:=3, x:=2*a+a*a;$ $a=?$, $x=?$
 - b) $x:=1, x:=x+11, x:=x*x-4;$ $x=?$
 - d) $a:=15, b:=a, a:=a-b;$ $a=?$, $b=?$
27. Quyidagi ko'rsatmalar ketma-ketligi qanday algoritm turiga misol bo'ladi? Algoritmlar natijasini aniqlang:
 - a) 1) $a:=3;$
 - 2) agar $a > 2$ bo'lsa, u holda $x:=2*a+a*a$ va 4-bandga o'tilsin;
 - 3) $x:=9-a*x;$
 - 4) javobi x yozilsin;
 - 5) tugatilsin;
 - b) 1) $x:=1;$
 - 2) agar $x > 2$ bo'lsa, u holda $x:=x+11$ va 4-bandga o'tilsin;
 - 3) $x:=x*x-4;$
 - 4) javobi x yozilsin;
 - 5) tugatilsin;
 - d) 1) $a:=15;$
 - 2) $b:=a;$
 - 3) agar $a > b$ bo'lsa, u holda $a:=a-b$ va 5-bandga o'tilsin;
 - 4) $a:=a+b;$
 - 5) javobi a, b yozilsin;
 - 6) tugatilsin.

II bob. IJROCHILAR VA ULARNING KO'RSATMALARI

Algoritmik tillar olamiga kirish

Ushbu qo'llanmaning asosiy maqsadi algoritmik tafakkurni rivojlantirish bo'lganligi bois, qo'yilgan masalalarni yechishda o'zimiz uchun dasturlash tiliga o'xshash va ularning umumiy jihatlarini o'z ichiga olgan maxsus tilni tashkil etamiz. Har qanday tilda bo'lgani kabi bu tilda ham **alifbo**, **sintaksisi** va **semantika** bo'ladi. Bu tushunchalarni qisqacha yoritib o'tamiz.

Alifbo – aniq bir til uchun asosiy belgilar ro'yxati, ya'ni shu tildagi matnlarni yozish uchun qo'llaniladigan «alifbo harflari» – boshqa belgini qo'llash mumkin emas.

Sintaksis – bu jummalarni hosil qilish qoidasi bo'lib, biror jumlani to'g'ri yoki xato yozilganligini aniqlash uchun xizmat qiladi. Aniqroq qilib aytadigan bo'lsak, til sintaksisi shu tilda belgilarni ma'noga ega bo'ladigan birlashtirishni aniqlab beruvchi qoidalar ro'yxati.

Semantika – jumla yoki gaplarning mazmunini aniqlaydi. Semantika hosil qilingan jumlar yoki gaplarni qanday amallar ketma-ketligini aniqlab berishini ta'minlaydi.

Qulay belgilashlarni o'ylab topish san'ati inson madaniyatida juda muhim ahamiyatga ega. Masalan, sonlarni belgilashni olaylik. Avvalgi bobda aytib o'tilganidek, hammangiz sonlarni ustun ko'rinishda qo'shish va ko'paytirishni bilasiz. Al-Xorazmiy tomonidan yoritib berilgan o'nlik belgilash sistemasi bunga imkon beradi. Rim raqamlari orqali yozilgan sonlarni qo'shib ko'ringchi, qo'shish masalasini hal etishda belgilashlar sizga hech qanday yordam bermayotganini ko'rasiz. Yana, masalan, musiqani olaylik. Musiqa tovushlarini belgilash uchun notani o'ylab topib, musiqachilar ancha murakkab va qiziqarli musiqalarni yozish hamda tarqatish imkoniyatiga ega bo'ldilar. Shu kabi juda ko'p misollarni keltirish mumkin.

Algoritmikada qulay belgilashlar ham, albatta, yanada muhim ahamiyatga ega. Bunda uning o'ziga xos tomonlari ham bor. Biz ham o'z belgilash usulimizni kiritishdan avval ba'zi tomonlarga

e'tibor berishimiz lozim bo'ladi. Yozish uchun lotin harflari va o'nlik sanoq sistemasidagi raqamlardan foydalanamiz. **Ko'rsatmalar** so'zlar bo'lib, ularni yozish uchun jumlalarga buyruq mazmunini beramiz. Bunda ularni qisqa va ma'noli bo'lishiga e'tibor beramiz. Misol uchun bir xil ma'noli quyidagi ko'rinishdagi jummalarni qarab chiqaylik:

o'ng tomonga yuring

o'ngga yuring

o'ngga yur

Bularning barchasini ko'rsatma sifatida qabul qilish mumkin. Lekin biz faqat **o'ngga yur** jumlasini tanlaymiz, ham qisqa, ham ma'noga ega, ham aniq amalni ifodalaydi.

Qisqa yozish chog'ida juda ham berilib ketish kerak emas. Juda qisqa belgilashlar tushunarsiz bo'lishi mumkin. Masalan, **o'ty** yoki **o'y** qanday ma'no anglatishini bilish qiyin. Doimo aql bilan murosali ish yuritish kerak.

Kompyuter uchun tushunarlik talabi ba'zan o'ta g'alati natijalarga olib keladi. Ba'zi kompyuter sistemalari probellar orasidagi har bir so'zni alohida tahlil qiladi. Ular uchun ko'rsatma ikkita so'zdan iborat bo'lishi mumkin emas, buni kompyuter ikkita alohida ko'rsatma deb hisoblashi mumkin. Bu kabi sistemalarda **o'nggayur** yoki **O'nggaYur** yoki **o'ngga_yur** ko'rinishidagi belgilashlarni uchratish mumkin.

E'tibor qiling — biz barcha hollarda ham ko'rsatma ichiga probel qo'ymaslikka harakat qildik! Ya'ni, agar siz biror joyda shunday belgilashlarni uchratsangiz, hayron bo'lmang. Bilingki, bularning hammasi texnik sabablarga asosandir. Kompyuter uchun shunisi qulay ekan.

Ijrochiga berilayotgan ko'rsatmalar mutlaqo ravshan bo'lishi kerak. Ular ikki xil ma'noli bo'lmasligi kerak. Oddiy inson tili juda ma'nodor, lekin ko'pincha bir ma'noli emas. Ba'zan bir so'zni o'zi ikki xil mazmunga ega bo'lishi mumkin, ba'zan esa ikki ma'nolilik jumlaning o'zida yashiringan bo'ladi. Masalan, quyidagi masala-hazilni ko'raylik.

Shoxda 7 ta qush o'tirgan edi. Ovchi 3 ta qushni otib tushirdi. Shoxda nechta qush qoldi?

Undagi «qoldi» so'ziga ikki xil mazmun berish mumkin. Birinchisi — tabiiyki, «tirik qoldi». Ikkinchisi — hazil — «joyida qoldi» (qolganlari uchib ketdi). Birinchi holda javob 4, ikkinchi holda javob 3. Masalani hazil ekanligini hisobga olsak, ikkinchi javob to'g'ri bo'ladi. Lekin kompyuter hazilni tushunmaydi.

ku! Bu kabi masalalarni yechishga kompyuterning kuchi yetmaydi.

Mana sizga go'zal qadimiy afsona. Uni tuzgan inson qalban dasturchi bo'lgan bo'lsa kerak.

Bir kuni Dionis hudosi shoh Midasga uning har qanday istagini bajarishni taklif etdi. Midas nimaga tegsa o'sha narsa oltinga aylanib qolishi istagini bildirdi. Bir necha soniyadan keyin u dunyodagi eng boy odamga aylandi. Lekin tezda kutilmagan qiyinchilik yuzaga keldi: shoh Midasning ovqati va ichimligi ham oltinga aylanib qolaverdi. Shoh ochdan o'lishi mumkin! Yaxshiki, Dionisning unga rahmi kelib o'zining bu sovg'asidan xolos etdi.

Ko'rib turibsizki, aniq ifodalash nafaqat kompyuter bilan ishlagandagina kerak. Albatta, Midas o'zining istagini aytayotganda ovqati va ichimligini ham oltinga aylanishini nazarda tutmagan. O'ylab ko'rib, balki yana bir qancha istisnolar qilishni xohlagan bo'lar edi. Hamma balo shundaki, Midas fikrini aniq va ravshan ifoda etmagan. U ularni o'z-o'zidan tushunarli deb hisoblab, faqat nazarda tutgan. Dastur tuzayotgan kishilar ko'pincha mana shunday holga tushib qolishadi.

Inson faoliyatining juda ko'p sohalarida aniqlik va ravshanlik muhimdir. Masalan, qonunlarni tuzishda, atom elektrostansiyasini yoki yadroli raketa qurilmasini boshqarish yo'riqnomasida hech qanday noaniqliklar, mujmal qismlar va ko'p ma'nolilik bo'lmasligi lozim. Bular butun sayyoramizning xavfsizligi bilan bog'liq! Qonun va yo'riqnomalar bilan bo'g'liq har bir ishda yetuk Algoritmik Tafakkur zarur.

Biroq boshqacha nuqtayi nazar ham mavjud. Qilishi kerak bo'lgan ish – yo'riqnomani aniq bajarish bo'lgani uchun atom bomba tugmasi yonida nima uchun inson o'tirishi kerak? Uni kompyuter bilan almashtirish mumkindek ko'rinadi. Baribir bu ishni insonga topshiramiz. Inson Ijrochi emas – u faqatgina buyruqlarni bajaribgina qolmay, balki mustaqil qaror qabul qiladi, javobgarlikni o'z zimmasiga oladi. Chamasi, dilimizda yo'riqnoma tugmani bosib atom urushini boshlashni talab qilganda Inson javobgarlikni o'z qo'liga oladi va yo'riqnoma buzadi, deb umid qilsak kerak.

Ba'zi mutaxassisliklarda har kuni va har daqiqada ko'rsatmalarni aniq tushunish zarur. Bularga samolyot uchuvchisi bilan aeroport dispetcheri (yoki kema kapitani va mashina bo'limi operatori) orasidagi muloqotni misol qilish mumkin. Agar dispetcher uzun mujmal jumtlar va ishoralar bilan gapirsa, u holda uchuvchi

(buni ustiga uni ko'rmayotgani uchun) yo'ldan adashadi va samolyotni halokatga uchratadi. Bu hol uchun odamlar quyidagicha yechimni topishdi. Ular avvaldan juda sodda, hattoki jo'n iboralar ro'yxatini tuzishdi va har bir ibora qanday ma'no anglatishini to'liq va aniq kelishib olishdi hamda biror-bir rasmiyatchiliksiz faqat shu iboralar yordamida gaplashishadi. Bu esa Ijrochining ko'rsatmalariga juda o'xshash, shunday emasmi? Chetdan qaraganda ular qandaydir o'zlarining tushunarsiz tilida gaplashayotgandek tuyuladi.

Kompyuterni yasaganlar ham shu kabi yo'ldan borishgan. Ular kompyuter bilan muloqot qilish uchun **dasturlash tillari** deb ataluvchi maxsus tillarni ishlab chiqishgan.

Siz dasturlash tillarini qanday qilib va nima uchun shundayligini tushunib olishingiz kerak. Shuning uchun qo'llanmada tilni hosil qilish yo'lini siz bilan birga yangitdan bosib o'tamiz. Ish davomida qabul qilingan qarorlar sababini tushuntirib boramiz.

Quyida biz kelgusida rioya qiladigan dastlabki qoidalarni sanab o'tamiz.

Sintaksis qoidalari:

- Ko'rsatma nomi kichik harflarda yoziladi.
- Har bir ko'rsatma alohida satrda yoziladi.

Ijrochi Dehqon

«Bo'ri, echki va karam» nomli qadimiy masala quyidagicha.

Dehqon daryoning chap qirg'og'ida bo'ri, echki va karam bilan turibdi. U bularni hammasini o'ng qirg'oqqa o'tkazishi kerak.

Uning qayig'i juda kichik bo'lgani uchun faqat bitta yo'lovchini olishi mumkin — yoki bo'rini, yoki echkini, yoki karamni. Yana — agar bo'ri va echki bir qirg'oqda qoldirilsa, bo'ri echkini yeb qo'yadi, agar echki va karamni bir qirg'oqda qoldirilsa, echki karamni yeb qo'yadi. Hayvonlar faqat dehqon borligidagina tinch turishadi. Qanday ish tutish kerak?

O'ylab ko'ring-chi, kimni birinchi bo'lib olib o'tish kerak? Tushunarliki, bo'rini olib bo'lmaydi - bu holda echki karam bilan bir qirg'oqda qoladi va uni yeb qo'yadi. Shu kabi sababga ko'ra karamni ham olib o'tib bo'lmaydi. Ammo echkini bimalol olib o'tish mumkin (bo'rilar odatda karamni yomon ko'rishadi). Shundan keyin dehqon bo'sh qayiq bilan qaytadi — echkini qaytarib olib kelishning ma'nosi yo'q.

Demak, masalamiz yechimining birinchi ikki qadami quyidagicha:
echkini o'tkaz
suzib o't

1-sharh

Yangi algoritmik tilni o'zimiz o'ylab topayotganimiz uchun bitta amalni bajarishni talab etuvchi ikkita yoki uchta so'z ham biz uchun bitta ko'rsatma bo'lishi mumkin.

Dehqon **echkini o'tkaz** ko'rsatmasiga binoan echkini o'tkazib qo'yadi, **suzib o't** ko'rsatmasiga binoan boshqa qirg'oqqa qayiqda suzib o'tadi.

Shundan keyin dehqonda ikkita imkoniyat bor:

- bo'rini o'tkazib qo'yish
- karamni o'tkazib qo'yish.

Agar u bo'rini o'tkazib qo'yib chap qirg'oqqa qaytsa, u holda bo'ri va echki o'ng qirg'oqda qoladi, bu esa echki uchun o'ta xavfli. Lekin bo'ri o'rniga karamni o'tkazib qo'yib chap qirg'oqqa qaytsa, o'ng qirg'oqda echki va karam yuqoridagi singari karam uchun xavfli holatda qoladi. Balki bu holatlar yechim yo'qligini bildirayotgandir.

Bu holatdan go'zal va kutilmagan g'oya qutqaradi. Ya'ni, masalan, bo'rini o'tkazib qo'yib orqaga bo'sh qaytmaymiz, o'zimiz bilan echkini olib o'tamiz! Keyingi qadamlar o'z-o'zidan ko'rinib turgani uchun masala yechimini ko'rsatmalar orqali yozishimiz mumkin:

echkini o'tkaz
suzib o't
bo'rini o'tkaz
echkini o'tkaz
karamni o'tkaz
suzib o't
echkini o'tkaz

2-sharh

Yuqoridagi ko'rsatmalar ketma-ketligidan ko'rinadiki, masalaning algoritmi chiziqli algoritm bo'lar ekan, ya'ni ko'rsatmalar kelish tartibida ketma-ket bajariladi.

3-sharh

Dehqon masalasida boshlang'ich qiymat: bo'ri, echki va karamni chap qirg'oqda ekanligi, natija esa bo'ri, echki va karamni o'ng qirg'oqda bo'lishi.

4-sharh

Bu masalada INKOR holat biror ko'rsatma bajarilganda yoki bo'ri echkini yoki echki karamni yeb qo'ysagina yuz beradi.

2.1-mashq

Dehqon o'ng qirg'oqqa echkini olib o'tgach, bo'rini emas, karamni olib o'tsin. Bu holda uning ko'rsatmalari ketma-ketligini yozing.

Ko'pincha bu masalani yechayotganda quyidagi kabi takliflar ham uchrab turadi:

- karamni bo'rining ustiga qo'yilsin (echki yaqinlashishga qo'rqishi uchun);
- bo'rini bir daraxtga, echkini esa boshqa daraxtga bog'lab qo'yilsin;
- karamni qayiqqa bog'lab qo'yilsin;
- echkini daryodan qayiqda emas, havo sharida o'tkazilsin;
- ko'prikdan o'tilsin va hokazo.

Albatta, masala shartida dehqon yaqinida ko'prik bor deb aytilmagan. Lekin shartda ko'prik yo'qligi haqida ham bir so'z ham aytilmagan!

Bularning hammasi to'g'ri. Masala shartida nima qila olishimiz va nima qila olmasligimiz haqida hech narsa deyilmagan. Masalan, dehqon qayiqni ortidan suzib hamda itarib harakat qilishi mumkinmi? Yoki karamni narigi qirg'oqqa irg'itishi mumkinmi? Bu kabi savollarga dehqonning bajarishi mumkin bo'lgan Ijrochi-dehqonning **ko'rsatmalar sistemasidagi** barcha amallari ro'yxatini sanab o'tish orqali javob berish to'g'ri bo'ladi. Ro'yxatda esa faqat to'rtta satr bor:

**bo'rini o'tkaz
echkini o'tkaz
karamni o'tkaz
suzib o't**

Dehqon esa faqat shu ro'yxatda ko'rsatilgan amallarnigina bajarishi mumkin, boshqa barcha amallar taqiqlanadi yoki boshqa barcha amallarni bajarish mumkin emas, chunki ular dehqonning ko'rsatmalar sistemasiga tegishli emas.

Masalani o'yin kabi ifodalash

Yuqoridagi masaladagi Dehqon o'zini g'alati tutmoqda. U haqiqiy hayotdagi oddiy dehqonlarga o'xshamaydi. Tirik dehqon kulishi, uxlashi, ot minishi, karamni yoqtirishi yoki yoqtirmasligi mumkin. Masalamizdagi dehqon esa bularning birortasini ham

bajarishga qodir emas (to'g'rirog'i, bularni yoki boshqa narsalarni bajara olishi mumkin, lekin bu bizni qiziqtirmaydi). U faqat to'rtta ko'rsatmani bajara olishi mumkin:

bo'rini o'tkaz
echkini o'tkaz
karamni o'tkaz
suzib o't

Hamma ham bu kabi sharoitga tushishi mumkin, masalan, shaxmat o'yinida. Shaxmat qoidalari shaxmatchi bajara oladigan imkoniyatlarni aniq ifodalaydi. Uning boshqa barcha odatlari (masalan, u karamni yoqtirishi) o'yin uchun ahamiyati yo'q.

Ammo bu yerda bir o'ta muhim farq bor. Shaxmatchi o'yin vaqtida o'zi qaror qabul qilishga majbur. Ijrochi esa mustaqil hech qanday qaror qabul qilmaydi. U faqat biz kiritgan ko'rsatmalarni bajaradi, xolos. Shu nuqtayi nazardan Ijrochiga Shaxmatchi emas, balki ko'proq Shaxmat Namoyishchisi o'xshaydi. Shaxmatchilar o'ynayotgan vaqtda tomoshabinlar o'yinni kuzatib borishi uchun Shaxmat Namoyishchisi katta magnit doskasidagi figuralarni siljitadi. Bu yerda Namoyishchi Shaxmatchilar berayotgan ko'rsatmalarni bajarayotgandek tuyuladi.

Tugmali har xil qurilmalar Ijrochiga ko'proq o'xshab ketadi. Elektron o'yinga o'xshash kichik qurilmalarni ko'z oldingizga keltiring. Ekran bo'ri, echki, karam va qayiqli dehqonni ko'rib turibsiz. Ekran yon tomonida to'rtta tugma bor. Birinchisida «bo'rini o'tkaz», ikkinchisida «echkini o'tkaz», uchinchisida «karamni o'tkaz», to'rtinchisida «suzib o't» yozuvi bor. Tugmalarni bosish orqali biz qahramonlarimizni ekranda siljitamiz. Ularni boshqarishning bundan boshqa yo'llari yo'q. Bu holda algoritumni yozish tugmalarni bosish ketma-ketligini tayinlash bilan barobardir.

Ko'rib turganingizdek, Dehqon haqidagi masalani o'yin deb hisoblashimiz mumkin. Ko'rsatmalar ro'yxatini tuzish orqali biz o'yin qoidasini aniqlashtirmoqdamiz, xolos. Tugmali qurilma misolida ko'rsatmalar ro'yxati — tugmalar to'plamidir.

Ijrochi Suvchi

Ijrochilarning yangi bir namunasini kiritamiz. Biz uni suv taqsimlash bilan band bo'lgani uchun Suvchi deb nomladik.

Bir litr suvni o'lchab olish qiyinmi? Javob bizda qanday hajmdagi idishlar borligiga bog'liq bo'ladi. Agar 1 litrli *A*

idishimiz bor bo'lsa, u holda algoritm bitta qadamdan iborat bo'ladi:

A ni to'ldir

Masalani ozgina qiyinlashtiramiz. Bizda ikki xil hajmli idish bor bo'lsin: 2 litrli *A* idish va 3 litrli *B* idish. U holda masala yechimi algoritmi ikki qadamdan iborat bo'ladi:

B ni to'ldir

B dan A ga quy

Birinchi qadamdan keyin *B* idishda 3 litr suv bo'ladi, ikkinchi qadamda biz 2 litr suvni *B* idishdan *A* idishga quyamiz, shundan keyin *B* idishda 1 litr suv qoladi.

2.1-masala

Bitta 3 litrli va bitta 5 litrli idish yordamida 1 litr suvni o'lchab oling.

Endi Suvchi va uning ko'rsatmalarini qat'iy tavsiflaymiz. Birinchi navbatda unda miqdori cheklanmagan suv manbai: daryo, ko'l yoki basseyn bo'lishi kerak. So'ngra idishlar soni (chelak, banka va hokazo) va ularning har birining hajmini aniq belgilab qo'yishimiz shart. Idishlarni lotin harflari, ya'ni *A*, *B*, *C*, ... bilan belgilaymiz. Suvchining ko'rsatmalari 3 xil bo'ladi.

Birinchi xili:

A (yoki B, C, ...) ni to'ldir

Bu ko'rsatmani bajarilishi natijasida mos idish chekkasiga suvga to'ladi. Bu kabi ko'rsatmalar idishlar soni nechta bo'lsa, shuncha bo'ladi.

Ikkinchi xili:

A (yoki B, C, ...) ni bo'shat

Bu ko'rsatmani bajarilishi natijasida mos idish bo'shatiladi. Bu kabi ko'rsatmalar soni idishlar soniga teng bo'ladi.

Va nihoyat, uchinchi xili:

A dan B (yoki A dan C va hokazo) ga quy

Bu ko'rsatma natijasi *B* hajmli idishdagi joy *A* hajmli idishdagi suvning barchasi uchun yetarli bo'lishiga bog'liq. Agar joy yetarli bo'lsa, *A* idish bo'shaydi, *B* idishdagi suv miqdori quyishdan oldin *A* va *B* idishlarda birgalikda qancha bo'lsa, shuncha bo'ladi. Agar joy yetarli bo'lmasa, u holda *B* idish to'la bo'ladi, *A* idishda esa *B* ga qancha sig'magan bo'lsa, shuncha suv bo'ladi. Uchinchi ko'rinishdagi ko'rsatmalar soni idishlar juftligi soniga teng.

Ikkita idishli Suvchining barcha ko'rsatmalarini yozib chiqamiz:

A ni to'ldir

B ni to'ldir

A ni bo'shat

B ni bo'shat

A dan B ga quy

B dan A ga quy

Agar idish soni 3 ta bo'lsa, barcha ko'rsatmalar soni $3 + 3 + 6 = 12$ ta.

2.2-mashq

Uchta A , B , C idishli Suvchining barcha ko'rsatmalari ro'yxatini yozib chiqing.

Odatda, har bir masalaning boshlanishida barcha chelaklar bo'sh bo'ladi. Bizning maqsadimiz biror-bir idishda kerakli miqdordagi suvni o'lchab olish: bunda qaysi chelakda bo'lishi va qolgan chelaklarda qancha suv qolishining ahamiyati yo'q.

2.2-masala

5 litrli A idish va bitta 8 litrli B idish bor.

- a) 1 litr suvni o'lchab oling.
- b) 4 litr suvni o'lchab oling.

Suvchi – bitta Ijrochi emas, o'xshash qoidalari ko'p Ijrochilardir. Ulardan birini ajratib olish uchun idish hajmini belgilab qo'yish zarur. Masalani qo'yish uchun esa qancha suvni o'lchab olishni xohlashingizni aytishingiz shart.

2.3-mashq

- a) O'zingizning Suvchi variantingizni o'ylang.
- b) Suvchingiz uchun masala tuzing.

5-sharh

Suvchi masalasida boshlang'ich qiymat: idishlarning bo'sh ekanligi, natija idishlarning birortasida talab etilgan suv miqdori bo'lishi.

Ijrochi Oshiruvchi

O'yin kabi aks ettirilgan yana bir Ijrochini quyidagicha tasvirlash mumkin: ekranida son aks etadigan avtomatik qurilmaning yon tomonida **1** ni **qo'sh** va **2** ga **ko'paytir** nomli ikkita tugma bor. 1-

tugmaning bosilishi ekrandagi sonni bittaga oshiradi, 2-tugmaning bosilishi ekrandagi sonni 2 ga ko'paytiradi. Avval ekranda 0 soni aks etib turadi.

Bu Ijrochini **Oshiruvchi** deb nomlaymiz.

Agar, masalan, ekranda 7 soni bo'lsa, va biz **1 ni qo'sh** tugmasini bossak, u holda 7 soni o'chib uni o'rnida 8 soni aks etadi. Agarda **2 ga ko'paytir** tugmasi bosilsa, 7 soni o'rnida 14 soni aks etadi.

Bizning tasavvurimizda Oshiruvchi ikkita ko'rsatmali ijrochidir:

1 ni qo'sh

2 ga ko'paytir

Ekranda 17 sonini hosil qilib ko'ramiz (0 dan boshlaymiz). Bu vazifani bajaradigan ikkita algoritmnini yozamiz:

1 ni qo'sh	1 ni qo'sh
1 ni qo'sh	2 ga ko'paytir
1 ni qo'sh	2 ga ko'paytir
1 ni qo'sh	2 ga ko'paytir
1 ni qo'sh	2 ga ko'paytir
1 ni qo'sh	1 ni qo'sh
1 ni qo'sh	
1 ni qo'sh	
1 ni qo'sh	
1 ni qo'sh	
1 ni qo'sh	
1 ni qo'sh	
1 ni qo'sh	
1 ni qo'sh	
1 ni qo'sh	
1 ni qo'sh	
1 ni qo'sh	
1 ni qo'sh	
1 ni qo'sh	

Bu algoritmlardan qaysi biri sizga ma'qul? Nima uchun?

2.4-mashq

17 sonini hosil qilish uchun o'zingiz biror algoritmni o'ylab toping. Bunday algoritm nechta ko'rsatmadan iborat bo'lishi mumkin:

- agar birinchi ko'rsatma **1 ni qo'sh** bo'lsa;
- umumiy holda.

2.3-masala

Nima deb o'ylaysiz, ekranda ixtiyoriy musbat sonni hosil qilish mumkinmi? Javobingizni asoslab bering.

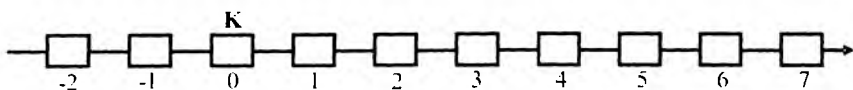
Quyidagicha musobaqa o'tkazing: kimdir son aytadi va barcha shu sonni hosil qilishga intiladi. Kim eng kam qadamli algoritm tuzsa, o'sha g'olib bo'ladi.

2.4-masala

- 15 sonini 8 tadan kam qadamda hosil qiling;
- 1024 sonini hosil qiling;
- Ekranga 4 soni yozilgan. Undan 15 sonini 6 tadan kam qadamda hosil qiling.

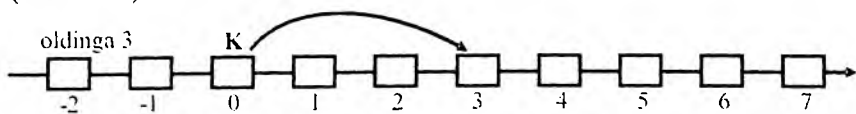
Ijrochi Chigirtka

Gorizontal to'g'ri chiziq o'tkazamiz va unda bir xil masofada kvadrat nishonlar joylashtiramiz (2.1-rasm). Nishonlar to'g'ri chiziqdagi nuqtalarni bildiradi. Nuqtalardan birini 0 bilan belgilaymiz. Undan o'ngdagi nuqtalarni 1, 2, 3, 4 va shu kabi, chapdagi nuqtalarni -1, -2, -3, -4 va shu kabi belgilaymiz. Bu to'g'ri chiziqni sonlar o'qi deb ataymiz. Sonlar o'qida Ijrochi **Chigirtka** yashaydi. Rasmda (va bundan keyin) u **K** harfi bilan belgilangan.

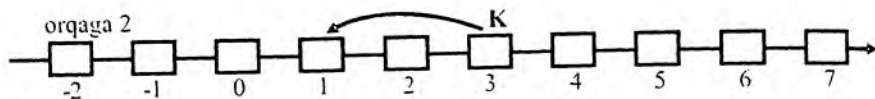


2.1-rasm.

Boshlang'ich vaqtda Chigirtka sonlar o'qining 0 nuqtasida joylashgan. U uch birlik oldinga (2.2-rasm) va 2 birlik orqaga (2.3-rasm) sakrashi mumkin.



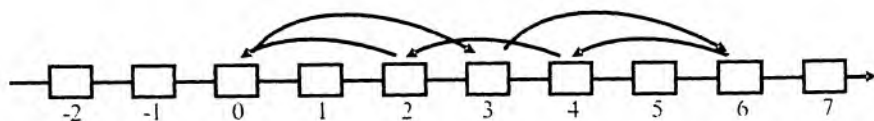
2.2-rasm.



2.3-rasm.

Quyidagi algoritm asosida Chigirtka qanday sakragani 2.4-rasmida ko'rsatilgan:

oldinga 3
oldinga 3
orqaga 2
orqaga 2
orqaga 2



2.4-rasm.

7-sharh

Chigirtka masalasida boshlang'ich qiymat: Chigirtkaning boshlang'ich holati, ya'ni qaysi tartib raqamli kvadratda turganligi bo'lsa, natija Chigirtkaning talab qilingan tartib raqamli kvadratga tushishidir.

2.5-mashq

- Chigirtkani 0 nuqtadan 7 nuqtaga o'tkazing.
- Chigirtkani 0 nuqtadan 2 nuqtaga o'tkazing.

2.5-masala

Ravshanda faqat 3 kg li toshlar bor, Rustamda esa faqat 2 kg li toshlar bor. Ravshan Rustamga 7 kg nok qarz. U qanday qilib shu toshlar yordamida qarzini uzadi?

Bu masala Chigirtka bilan biror bog'liqlikka egami?

2.6-masala

Chigirtkani 0 dan 5 gacha kesmadan chiqmagan holda 2, 3, 4 va 5 nuqtalarning har birida bir martadan bo'lishga majbur qiling.

2.7-masala

Chigirtka algoritmini bajarishni 0 nuqtada boshladi va 2 nuqtada tugatdi. Shu algoritm yana bir marta bajarildi. Chigirtka qaysi nuqtaga borib qoldi?

2.8-masala

Chigirtka to'g'ri chiziqda yotgan ixtiyoriy nuqtaga bora oladimi? Qanday qilib? (Albatta, biz kasr sonlar emas, faqat butun sonlarga mos nuqtalar haqida so'rayapmiz.)

2.9-masala

Chigirtkaning ko'rsatmalar ro'yxatini ozgina o'zgartiramiz. Albatta, ko'rsatmalar ro'yxatini ozgina o'zgartirilishi yangi Ijrochi hosil bo'lganligini bildiradi. Lekin baribir uni ham Chigirtka deb atayveramiz. Bu Chigirtka uchun yangi ko'rsatmalar ro'yxati ham ikkita ko'rsatmadan iborat bo'ladi:

oldinga 7

orqaga 5

Yangi Chigirtka to'g'ri chiziqda yotgan ixtiyoriy nuqtaga, masalan, 1 nuqtaga bora oladimi?

E'tibor qilgan bo'lsangiz, bizda idishlarining soni va hajmi bilan farqlanadigan juda ko'p har xil Suvchi bor edi. Xuddi shunday turli Chigirtkalar bilan ish ko'rishimizga to'g'ri keladi: ular oldinga va orqaga har turli sakrashlarni bajaradi.

Algoritmlashni qanday yengillashtish mumkin

Yana Suvchi masalasiga qaytamiz. Bizda 5 litrli A idish va 8 litrli B idish bor, 4 litr suvni o'lchab olish kerak. Masalaning yechimi quyidagicha bo'ladi:

A ni to'ldir

A dan B ga quy

A ni to'ldir

A dan B ga quy

B ni bo'shat

A dan B ga quy

A ni to'ldir

A dan B ga quy

A ni to'ldir

A dan B ga quy

Bu yerda nima yozilgani tushunarlimi? Bizningcha, yo'q. Algoritmida 10 ta ko'rsatma bo'lgani bilan o'zimiz ham tushunmayapmiz. Agar algoritmida ular 1000 ta bo'lsa-chi? U holda biz ular ichida adashib ketamiz va bu barcha quyilayotgan suvda cho'kib ketamiz.

Agar bir algoritmni yozilishini yengillashtirmoqchi bo'lsak va to'g'ri ishlayotganiga amin bo'lmoqchi bo'lsak, u holda bizga uni tekshirish uchun qadamba-qadam boshidan-oxirigacha vosita kerak bo'ladi.

Baxtimizga qanday vosita ekanligi ravshan. Balki siz bu bobni o'qib chiqqunga qadar mustaqil ravishda ixtiro etgan bo'lishingiz mumkin. Bizga barcha obyektini har bir ko'rsatma bajarilgandan keyingi holatini yozib chiqish yetadi, xolos. Bizning masalamizda bu holatlar faqatgina har bir idishdagi suvning miqdoridir.

Shunday jadval tuzamiz:

Ko'rsatma	A [5 litr]	B [8 litr]
	0	0
A ni to'ldir	5	0
A dan B ga suv	0	5
A ni to'ldir	5	5
A dan B ga suv	2	8
B ni bo'shat	2	0
A dan B ga suv	0	2
A ni to'ldir	5	2
A dan B ga suv	0	7
A ni to'ldir	5	7
A dan B ga suv	4	8

Eng so'nggi holat: A idishda – 4 litr suv.

Endi har bir ko'rsatmani alohida va boshqalaridan mustaqil osongina tekshirish mumkin. Masalan, 4-ko'rsatmani tekshiramiz. Uni bajarilishidan avval har bir idishda 5 litrdan suv bor edi; demak, A idish to'la, B idishda esa 3 litr hajmdagi bo'sh joy bor. Bundan kelib chiqadiki, A dan B ga 3 litr suv quya olamiz, A da esa 2 litr suv qoladi va B idish to'ladi (ya'ni, undagi suv 8 litr bo'ldi).

2.6-masha

Algoritmni hamma qadamini tekshirib chiqing.

2.10-masala

Uchta ritsar uchta qurolbardorlari bilan daryo bo'yiga kelishdi. Daryo bo'yida ikki kishi suza oladigan qayiqchani topib olishdi. Har bir qurolbardor xo'jayiniga shunchalik sodiqki, u boshqa ritsarlar bilan xo'jayinisiz yolg'iz o'zi qirg'oqda qolishga yoki qayiqda suzishga rozi bo'lmayapti. Lekin ular yakka o'zi yoki boshqa qurolbardorlar bilan qolishga rozi. Olti kishilashib ular daryoni qanday suzib o'tishadi?

Bu masaladagi Ijrochi uchun ko'rsatmalar tizimini ishlab chiqing. Shu ko'rsatmalar tizimida masalaning yechimini beruvchi algoritmnini tuzing. Algoritmnini bajarilishidagi holatni ifodalovchi jadval tuzing va uni to'ldiring.

Quyidagi uch qoidaga amal qilinishi kerakligini eslatib o'tamiz:

- qayiq ikki kishidan ortig'ini ko'tarmaydi;
- qayiqda qurolbardor begona ritsar bilan qolishi mumkin emas;
- agar qirg'oqda qurolbardor bilan begona ritsar birga bo'lsa, u holda bu qurolbardorning ritsari ham shu yerda bo'lishi shart.

Birinchi ikkita qoidaning bajarilishini nazorat qilib turish uchun algoritmnini ko'rish kifoya.

Lekin oxirgi qoida buzilmaganiga ishonch hosil qilish uchun jadval kerak bo'ladi.

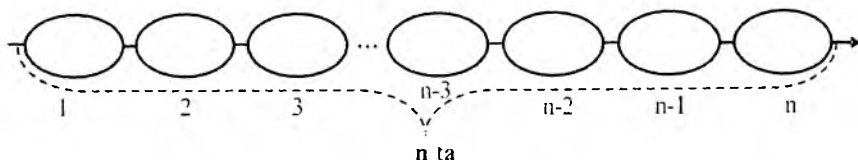
Malakali dasturchilar dastur yozayotganda ichiga **dastur holati haqidagi tasdiqlarni** joylashtirishadi. Ular dasturda talab qilinganidek, ishlayotganiga ishonch hosil qilishga yordam beradi va **dasturning to'g'riligini isbotlashga** imkon beradi.

Ba'zan bu malol keladigandek tuyuladi. Lekin dasturchilar mehnati eng hurmatli ishlardandir. Hozirgi kunda kompyuterlar samolyotlar va atom elektrstansiyalarni boshqarmoqda, vrachlarga jarrohlik operatsiyalarida yordam bermoqda, juda ko'p har xil muhim ishlarni bajarmoqda. Inson va insoniyatning hayoti, atrof-muhitning holati, sarflangan ulkan mablag'larning natijasi kompyuterlarning ishi bilan bog'liq ekan, undagi dasturlar to'g'ri ishlayotganiga ishonch hosil qilish shart. Bunday ishonch hosil qilishning eng mukammal usullaridan biri – dasturlash fanidir.

Yuqorida keltirilgan mashq va masalalarni yechib, ya'ni yechish algoritmining ko'rsatmalarini yozib chiqib, bu algoritmlar chiziqli ekanligini ko'rishingiz mumkin.

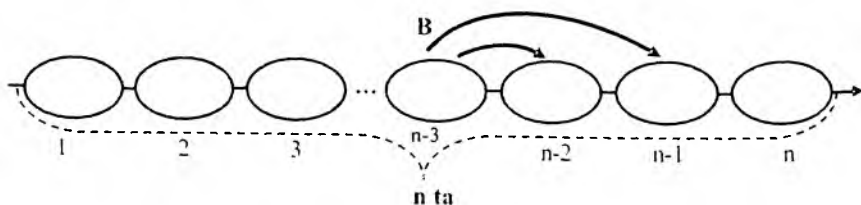
Ijrochi Baqa

Horizantal to'g'ri chiziq o'tkazamiz va unda bir xil masofada 1 dan n gacha tartiblangan ellipsli nishonlar joylashtiramiz (2.5-rasm). Ellipsli nishonlar suv ustida turgan nilufar barglarini bildiradi. Bu barglar ustida Ijrochi **Baqa** ovqatlanadi. Baqani bundan keyin **B** harfi bilan belgilaymiz.

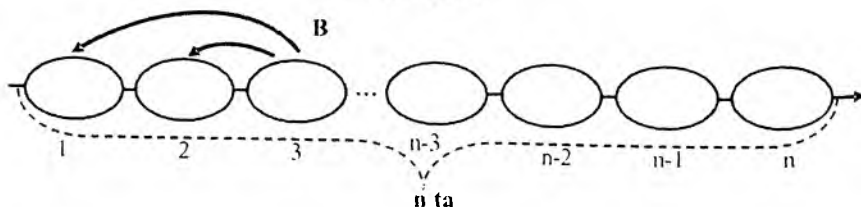


2.5-rasm.

Har kuni tongda nilufar barglari ustida bittadan pashsha o'tirgan bo'ladi. Baqa tongda ovqatlanish uchun uyidan a tartib raqamli bargni ustiga tushadi. U 1 yoki 2 birlik oldinga (2.6-rasm) va 1 yoki 2 birlik orqaga (2.7-rasm) sakrashi mumkin. Baqa barglar ustida sakrab barcha pashshani yeb tugatishi kerak. Lekin barglar shunchalik nozikki, Baqa ular ustidagi pashshani yeb bo'lgach, barglar cho'kib ketadi. Sakrashlar natijasida Baqa b tartib raqamli bargni ustiga borishi kerak, undan uyiga chiqib keta oladi.



2.6-rasm.



2.7-rasm.

8-sharh

Baqa masalasida boshlang'ich qiymat: Baqa a tartib raqamli bargda joylashishi, natija Baqaning b tartib raqamli bargga kelishi ekan.

9-sharh

Baqa masalasida INKOR holat ko'rsatma Baqani cho'kib ketgan bargga sakrashga majbur qilganida yoki 1 dan kichik yoki n dan katta tartib raqamli bargga sakrashga majbur qilganida ro'y beradi.

Demak, Baqaning barcha ko'rsatmalari ro'yxati quyidagilardan iborat bo'ladi:

oldinga 1

oldinga 2

orqaga 1

orqaga 2

2.7-masho

Baqa 10 ta bargli nilufarning 5 tartib raqamli bargi ustiga tushib, yuqoridaagi ko'rsatmalar ro'yxatini algoritm hisoblab INKOR holatisiz baiara olishi mumkinmi? Mumkin bo'lmasa, ularning o'rnini almashtirish orqali mumkin holatga keltiring. Sababini rasm orqali izohlang.

2.11-masala

Baqa 5 ta bargli nilufarning 3 tartib raqamli bargiga tushdi. U quyidagi algoritmni bajarib, qaysi tartib raqamli barg ustiga borishini va qaysi bargdagi pashsha yeyilmay qolishini aniqlang:

orqaga 2

oldinga 1

oldinga 2

2.8-masho

Yuqoridaagi masalada ikkinchi ko'rsatma **orqaga 1** ko'rsatmasi bilan almashtirilsa, qanday natija olinishini aniqlang.

2.12-masala

Baqa 5 ta bargli nilufarning 3 tartib raqamli bargida turibdi. U quyidagi algoritmni bajarganda yeyilmay qolgan pashshani yeyishi uchun ko'rsatma qo'shing:

orqaga 2

oldinga 1

oldinga 2

2.13-masala

Baqa 5 ta bargli nilufarning 3 tartib raqamli bargida turibdi. U 4 tartib raqamli barg ustiga borish algoritmini tuzing.

2.14-masala

Baqa 4 ta bargli nilufarning 2 tartib raqamli bargida turibdi. U 3 tartib raqamli barg ustiga bora oladimi? Agar javob ijobiy bo'lsa, u holda borish algoritmini tuzing, aks holda INKOR deb yozing.

Nazorat savollari va topshiriqlar

1. Tilning alifbosi deƣanda nima tushuniladi?
2. Tilning sintaksisi va semantikasi deƣanda nima tushuniladi?
3. Ko'rsatmalar aanda xususiyatlarƣa eƣa bo'lishi kerak?
4. Iirochi Dehaon uchun ko'rsatmalar sistemasi haoida so'zlab berinƣ.
5. Bo'ri va echki karam masalasi yechimini blok-sxema ko'rinishida tasvirlanƣ.
6. Iirochi Suvchi uchun ko'rsatmalar sistemasi haoida so'zlab berinƣ.
7. Iirochi Oshiruvchi uchun ko'rsatmalar sistemasi haoida so'zlab berinƣ.
8. Iirochi Chiƣirtkaninƣ iirochi muhiti haoida so'zlab berinƣ.
9. Iirochi Chiƣirtka uchun ko'rsatmalar sistemasi haoida so'zlab berinƣ.
10. Iirochi Baaninƣ iirochi muhiti haoida so'zlab berinƣ.
11. Iirochi Baaa uchun ko'rsatmalar sistemasi haoida so'zlab berinƣ.
12. Bobda keltirilƣan barcha Iirochilar uchun sodda amallarni vozib chidinƣ.
13. Bobda keltirilƣan barcha Iirochilar uchun boshlanƣ'ich holat va natija haoida so'zlab berinƣ.
14. Bobda keltirilƣan barcha Iirochilar uchun bor bo'lsa kamida bitta INKOR holatni vozinƣ.
15. Bobda keltirilƣan barcha mashalarni baiarinƣ.

Qo'shimcha masalalar

Suvchi uchun

S-2.1. Quvidaƣi algoritmninƣ har bir ko'rsatmasi baiirilƣandan kevin har bir idishdaƣi suv miadorini vozinƣ:

Ko'rsatma	A [4 litr]	B [11 litr]
	0	0
A ni to'ldir		
A dan B oa quv		
A ni to'ldir		
A dan B oa quv		
A ni to'ldir		
A dan B oa quv		
A ni to'ldir		
A dan B oa quv		

S-2.2. Quvida *A* va *B* idishlarni Suvchi qandavdir algoritmni bajaravotgandagi holati keltirilgan:

Ko'rsatma	A [4 litr]	B [7 litr]
	0	0
	0	7
	4	3
	0	3
	3	0
	3	7
	4	6
	0	6
	4	2

Shu algoritm ko'rsatmalarini qavta tiklang. Algoritmni shundav to'ldirishga harakat olingki, uning ishlashi natijasida idishlardan birida 1 litr suv qolsin.

S-2.3. Yangi algoritmdagi yetishmayotgan ko'rsarmalarni va idishlardagi suv miqdorini qavta tiklang:

Ko'rsatma	A [7 litr]	B [17 litr]
	2	17
	0	12
B dan A ga quv		
B ni bo'shat		
	5	17
B dan A ga quv		
	7	8
	0	1

S-2.4. *A* idishning haimi 3 litrga, *B* idishning haimi esa 2 litrga teng bo'lsin.

a) Ikki ta ko'rsatmadan iborat shunday algoritm tuzingki, uni bajarish natijasida *A* idishda 1 litr suv qolsin;

b) *A* idishda 1 litr suv qolishi kerak bo'lsa, bu algoritm yana qanday haimdagi idishlar uchun yaravdi?

d) Shunday algoritm tuzingki, uni bajarish natijasida *B* idishda 1 litr suv qolsin.

S-2.5. 9 litrli va 2 litrli idishlari bor suvchi uchun 1 litr suvni o'lchab olishi mumkin bo'ladigan algoritm tuzing.

S-2.6. 5 litrli va 3 litrli idishlari bor suvchi uchun 1 litr suvni o'lchab olishi mumkin bo'ladigan algoritm tuzing. Algoritmni eng qisqa bo'lishiga harakat qiling.

S-2.7. 7 litrli va 5 litrli idishlari bor suvchi uchun 1 litr suvni o'lchab olishi mumkin bo'ladigan algoritm tuzing. Algoritmni eng qisqa bo'lishiga harakat qiling.

S-2.8. 6 litrli va 4 litrli idishlari bor suvchi uchun 1 litr suvni o'lchab olishi mumkinmi? Javobingizni izohlab bering.

S-2.9. Suvchi 2 litrli *A* idish oldi. Suvchi 1 litr suvni o'lchab olishi mumkin bo'lishi uchun *B* idishning haimi qanday bo'lishi kerak? Bu haimning barcha qiymatlarini qanday vozib ko'rsatish mumkin? Javobingizni izohlab bering.

S-2.10. Suvchi 4 litrli *A* idish oldi. Suvchi 1 litr suvni o'lchab olishi mumkin bo'lishi uchun *B* idishning haimi qanday bo'lishi kerak? Bu haimning barcha qiymatlarini qanday vozib ko'rsatish mumkin? Javobingizni izohlab bering.

S-11. Suvchi 3 litrli *A* idish oldi. Suvchi 1 litr suvni o'lchab olishi mumkin bo'lishi uchun *B* idishning haimi qanday bo'lishi kerak? Bu haimning barcha qiymatlarini qanday vozib ko'rsatish mumkin? Javobingizni izohlab bering.

S-12. *A* idishning haimi 6 litr, *B* idishning haimi 9 litr. Suvchi 1 litr suvni o'lchab olishi mumkinmi?

S-13. Suvchi 6 litrli *A* idish oldi. Suvchi 1 litr suvni o'lchab olishi mumkin bo'lishi uchun *B* idishning haimi qanday bo'lishi kerak? Bu haimning barcha qiymatlarini qanday vozib ko'rsatish mumkin? Javobingizni izohlab bering.

S-14. «*A* va *B* idishlar vordamida Suvchi 1 litr suvni o'lachab olishi mumkin bo'lishi uchun ularning hajmlari o'zaro tub (1 dan farali umumiy bo'luvchi ga ega emas) bo'lishi kerak" xulosani to'g'riligini tekshiring.

S-15. *A* idishning hajmi 16 litr, *B* idishning hajmi 5 litr. Quvidagilarni qanday o'lachab olish mumkin:

- a) 9 litr;
- b) 8 litr?

S-16. Suvchida 3 ta hajmi 6 litrli, 12 litrli va 15 litrli idishlar bor. U quvidagilarni o'lachab olish mumkinmi:

- a) 1 litr;
- b) 2 litr;
- d) 3 litr?

S-17. Suvchida 3 ta hajmi 6 litrli, 10 litrli va 15 litrli idishlar bor. U quvidagilarni o'lachab olish mumkinmi:

- a) 1 litr;
- b) 7 litr?

Agar mumkin bo'lsa, mos algoritmlarni vozing.

S-18. Suvchida 4 litr sut solingan 6 litrli *A* idish, 1 litli bo'sh *B* idish va 4 litr suv solingan 6 litrli *C* bor. Suvchi quvidagi algoritmni bajardi:

A dan B ga o'uv
B dan C ga o'uv
C dan B ga o'uv
B dan A ga o'uv

U bu ishlarni shunday tezlikda bajardiki, hech kim sezmay qoldi: ikkinchi ko'rsatmadan kevin sut suvga bir xil aralashdimi? Nima ko'p – sut suvdami voki suv suvdami? Javobingizni izohlab bering.

Oshiruvchi uchun

O-2.1. Oshiruvchi ish boshlashidan avval ekranda 1 vonib turgandi. Oshiruvchi quvidagi algoritmni bajardi:

2 ga ko'pavtir
2 ga ko'pavtir
1 ni o'sh
2 ga ko'pavtir

a) Oshiruvchi bu algoritmni bajaraganidan kevin ekranda qanday son vonib turadi?

b) Oshiruvchi nechta ko'rsatma bajardi?

O-2.2. Oshiruvchi ish boshlashidan avval ekranda 5 vonib turgandi Oshiruvchi quyidagi algoritmni bajaradi:

2 ga ko'navtir

2 ga ko'navtir

1 ni o'sh

2 ga ko'navtir

Oshiruvchi bu algoritm bajaraganidan kevin ekranda qanday son vonib turadi? Oshiruvchi har bir ko'rsatmani bajaraganidan kevin ekranda qanday son vonib turadi?

O-2.3. Oshiruvchi faqat **1 ni o'sh** ko'rsatmasini bajarib, 0 sonidan 13 sonini hosil qildi. Oshiruvchi nechta ko'rsatma bajaragan?

O-2.4. Oshiruvchi faqat **1 ni o'sh** ko'rsatmasini bajarib, 27 sonidan 293 sonini hosil qildi. Oshiruvchi nechta ko'rsatma bajaragan?

O-2.5. Oshiruvchi 33 sonidan 19 sonini hosil qilishi mumkinmi?

O-2.6. Oshiruvchi har qanday sondan undan katta ixtiyoriv qanday sonni hosil qilishi mumkinligini isbotlang.

O-2.7. Oshiruvchi ish boshlashidan avval ekranda 1 vonib turgandi. Oshiruvchi faqat **2 ga ko'navtir** ko'rsatmasini 10 marta bajaradi. Har bir ko'rsatmadan kevinigi Oshiruvchi ekranidagi holatni vozib chiqing. Algoritm ishi bajarilgandan kevin ekranda qanday son vonib turadi?

O-2.8. Oshiruvchi faqat **2 ga ko'navtir** ko'rsatmasini bajarib, 2 sonidan 512 sonini hosil qildi. Oshiruvchi nechta ko'rsatma bajaragan?

O-2.9. Oshiruvchi faqat **2 ga ko'navtir** ko'rsatmasini bajarib, 12 sonidan 192 sonini hosil qildi. Oshiruvchi nechta ko'rsatma bajaragan?

O-2.10. Oshiruvchi faqat **2 ga ko'navtir** ko'rsatmasini bajarib, 15 sonidan 280 sonini hosil qilishi mumkinmi?

O-2.11. 1 sonidan 8 sonini hosil qilish uchun Ravshan quyidagi algoritmni vozdi:

2 ga ko'navtir

1 ni o'sh

1 ni o'sh

2 ga ko'navtir

Shu natijani beruvchi eng aysaa algoritmni vozinq.

O-2.12. 1 sonidan 12 sonini hosil qilish uchun quyidagi algoritmni vozishdi:

2 ga ko'navtir

2 ga ko'navtir

2 ga ko'navtir

1 ni oo'sh

1 ni oo'sh

1 ni oo'sh

1 ni oo'sh

Shu natijani beruvchi eng aysaa algoritmni vozinq. Algoritmingizda eng oxirgi ko'satma qandav?

O-2.13. 5 sonidan 105 sonini hosil qilish uchun Odil quyidagi algoritmni vozdi:

2 ga ko'navtir

1 ni oo'sh

1 ni oo'sh

2 ga ko'navtir

1 ni oo'sh

2 ga ko'navtir

1 ni oo'sh

1 ni oo'sh

2 ga ko'navtir

1 ni oo'sh

Odilning algoritmi shu natijani beruvchi eng aysaa algoritmmi? Agar vo'a bo'lsa, u holda shu natijani beruvchi eng aysaa algoritmni vozinq. Algoritmingizda eng oxirgi ko'satma qandav?

O-2.14. Ekranda 8 vonib turgandi. 18 sonini 8 dan quyidagi 3 ta ko'rsatma hosil qiladi:

2 ga ko'navtir

1 ni oo'sh

1 ni oo'sh

Lekin bundan ham tez hosil qilish mumkin. Eng aysaa algoritmni vozinq.

Chigirtka uchun (oldinga 3 va orqaga 2 ko'rsatmali hol)

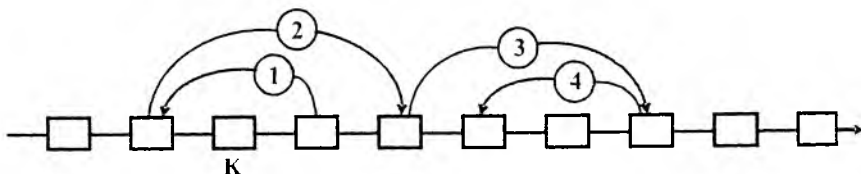
Ch-2.1. Quyidagi algoritmni bajarqan Chigirtkaning ko'chishlarini strelka bilan ko'rsating (Chigirtkaning boshlang'ich holati ixtivoriy):

oldinga 3

orqaga 2

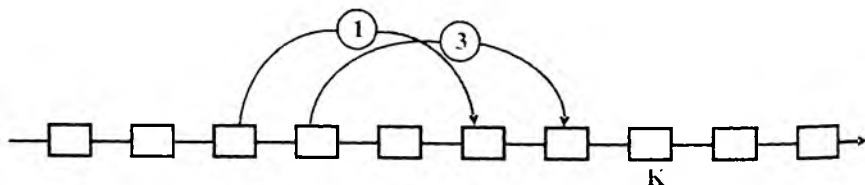
oldinga 3
oldinga 3
oraqaga 2
oraqaga 2

Ch-2.2. 2.5-rasmda qandavdir algoritmni bajarган Chigirtkaning ko'chishlarini strelka bilan ko'rsatilgan. Shu algoritmni tiklang.



2.5-rasm.

Ch-2.3. Chigirtka bajarган 5 ta ko'rsatmali algoritmning 5-ko'rsatmasi **oraqaga 2** edi. 2.6-rasmda strelka bilan 1 va 3-ko'rsatmalarni qandav bajarгани va algoritm bajarilgandan kevin qaverga borгани ko'rsatilgan.

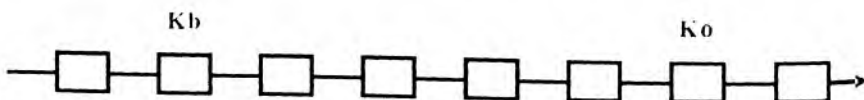


2.6-rasm.

Shu algoritmni tiklang.

Ch-2.4. Chigirtka quyidagi algoritmni bajardi:

oldinga 3
oraqaga 2
 ...
oraqaga 2
oldinga 3



2.7-rasm.

2.7-rasmda strelka bilan Chiğirtka algoritmi ko'rsatmalarini qanday bajarilganini ko'rsatilgan edi, strelkalar tasodifan o'chib ketdi. Aksiga olib, 3-ko'rsatma ham o'chib ketdi. Chiğirtkaning boshlang'ich holati rasmda Kb bilan oxirgi holati Ko bilan ko'rsatilgan. Qavsi ko'rsatma o'chib ketgan?

Ch-2.5. Chiğirtka bajarilgan oquvdaagi algoritmdan ikkita ko'rsatma o'chib ketdi:

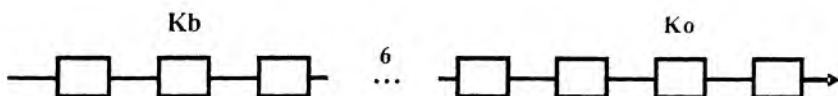
oldinga 3

...

...

oraqaga 2

oldinga 3



2.8-rasm.

2.8-rasmda Chiğirtkaning boshlang'ich holati Kb bilan oxirgi holati Ko bilan ko'rsatilgan. Shu ko'rsatmalarni tiklang.

Ch-2.6. Chiğirtkani 0 tartib raqamli kvadratdan 4 tartib raqamli kvadratga o'tkazuvchi 3 ta ko'rsatmadan iborat algoritmi tuzing.

Ch-2.7. Chiğirtkani 0 tartib raqamli kvadratdan -1 tartib raqamli kvadratga o'tkazuvchi 3 ta ko'rsatmadan iborat algoritmi tuzing.

Ch-2.8. Chiğirtkani 0 tartib raqamli kvadratdan -3 tartib raqamli kvadratga o'tkazuvchi 4 ta ko'rsatmadan iborat algoritmi tuzing.

Ch-2.9. Chiğirtkani 0 tartib raqamli kvadratdan 7 tartib raqamli kvadratga o'tkazuvchi 4 ta ko'rsatmadan iborat algoritmi tuzing.

Ch-2.10. Har biri Chiğirtkani 0 tartib raqamli kvadratdan -1 tartib raqamli kvadratga o'tkazuvchi uchta ko'rsatmadan iborat uchta turli algoritmi tuzing.

Ch-2.11. Har biri Chiğirtkani 0 tartib raqamli kvadratdan 4 tartib raqamli kvadratga o'tkazuvchi uchta ko'rsatmadan iborat uchta turli algoritmi tuzing.

Ch-2.12. Chiğirtka 0 tartib raqamli kvadratdan 7 tartib raqamli kvadratga to'rtta sakrashda o'tdi.

- a) Bu ishni necha xil algoritmlar yordamida bajarish mumkin?
 b) Bu algoritmlardan nechtasini bajarish jarayonida Chigirtka manfiy tartib raqamli kvadratga tushmaydi?

Ch-2.13. Chigirtka 0 tartib raqamli kvadratdan chiqib, quyidagi algoritmni bajardi:

oroqaga 2
oldinga 3
oldinga 3
oldinga 3
oroqaga 2
oroqaga 2
oroqaga 2

Chigirtka bo'lib o'tgan kvadratlarning tartib raqamlarini vozib chiqing.

Ch-2.14. Chigirtka 0 tartib raqamli kvadratdan chiqib, quyidagi algoritmni bajarib 0; 3; 1; 4; 2; 0; -2 tartib raqamli kvadratlarda bo'ldi.

a) Agar Chigirtkaning boshlang'ich iovi 5 tartib raqamli kvadratda bo'lsa, xuddi shu algoritmni bajaranda qaysi kvadratlarda bo'lishini vozib chiqing.

b) Algoritmni to'liq tiklang.

Ch-2.15. Quyidagi algoritmni bajaragan Chigirtka -2 tartib raqamli kvadratdan 6 tartib raqamli kvadratga o'tdi. Agar chigirtkaning boshlang'ich iovi 1 tartib raqamli kvadratda bo'lsa, xuddi shu algoritmni bajaranda qaysi kvadratga o'tadi?

Ch-2.16. Chigirtka algoritmning 5 ta ko'rsatmasini bajarib, vana harakatini boshlagan holatiga qaytib keldi.

a) Shunday ish bajaradigan algoritmlar tuzing;

b) Shu ishni bajaradigan nechta algoritmlar bor?

Ch-2.17. Chigirtka algoritmlar ko'rsatmalarini bajarib, vana harakatini boshlagan holatiga qaytib keldi. Algoritmning **oroqaga 2** ko'rsatmasi **oldinga 3** ko'rsatmasidan 10 marta ko'p ekan. Algoritmning nechta ko'rsatma bo'lgan?

Ch-2.18. Chigirtka **oroqaga 2** ko'rsatmasi **oldinga 3** ko'rsatmasidan 13 marta ko'p bo'lgan algoritmni bajardi. Chigirtka nechta kvadratga surildi va qaysi vo'nalishda?

Ch-2.19. Chigirtka quyidagi algoritmni bajarib, harakatini boshlagan holatidan 3 ta kvadrat chiqqan surilib oldi. Algoritmning hammasi bo'lib 13 ta ko'rsatma bor.

Ch-2.20. Chigirtka qandavdir algoritmni bajarib, harakatini boshlagan holatidan 6 ta kvadrat o'ngga surilib qoldi. Algoritmدا hammasi bo'lib 17 ta ko'rsatma bor. Algoritmدا nechta **orqaga 2** ko'rsatmasi **oldinga 3** ko'rsatmasi bor?

Baqa uchun

B-2.21. Quvidagi algoritmni bajaragan Baqani ko'chishlarini strelka bilan ko'rsating (Baqani boshlang'ich holati ixtiyoriv):

oldinga 2

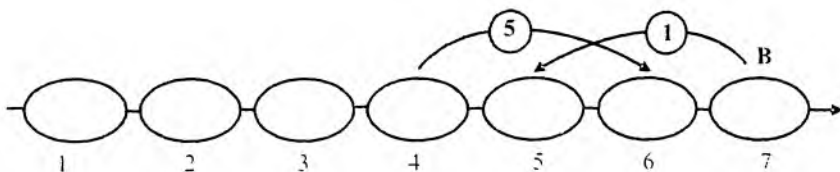
orqaga 1

oldinga 2

oldinga 2

orqaga 1

B-2.22. Baqa bajaragan 5 ta ko'rsatmali algoritmning 3-ko'rsatmasi **orqaga 1** edi. 2.9-rasmda strelka bilan 1- va 5-ko'rsatmalarni qandav bajarгани va algoritm bajarilgandan kevin qaverga borгани ko'rsatilgan.



2.9-rasm.

Shu algoritmni tiklang.

B-2.23. Baqani 3 tartib raqamli bargdan avval 6, kevin 4 tartib raqamli bargga o'tkazuvchi 3 ta ko'rsatmadan iborat algoritm tuzing.

B-2.23. Baqani 1 tartib raqamli bargdan 7 tartib raqamli o'tkazuvchi uchta ko'rsatmadan iborat uchta turli algoritm tuzing.

B-2.23. Har biri Baqani 3 tartib raqamli bargdan 7 tartib raqamli bargga o'tkazuvchi 3 ta ko'rsatmadan oshmaydigan uchta algoritm tuzing.

B-2.24. Baqa 1 tartib raqamli bargdan 7 tartib raqamli bargga o'rtta sakrashda o'tdi. Bu ishni nechta xil algoritm yordamida bajarish mumkin?

B-2.25. Baga 1 tartib raqamli bargdan chiqib va qandavdir algoritmni bajarib 3; 2; 4; 5; 7; 6; 8 tartib raqamli barglarda bo'ldi.

a) Algoritmni to'liq tiklang.

b) Agar Baganing boshlang'ich iovi 5 tartib raqamli bargda bo'lsa, xuddi shu algoritmni bajaranda qavsi tartib raqamli barglarda bo'lishini vozib chiqing.

B-2.26. Baga algoritm ko'rsatmalarini bajarib harakatini boshlagan holatidan bitta chaddagi bargga qavtib keldi. Baga 25 tartib raqamli bargda bo'lgani ma'lum bo'lsa, eng kam qadamli algoritmni tiklang.

B-2.27. Baga bitta **oldinga 1** ko'rsatmasi bor, **orqaga 2** ko'rsatmasi oldinga 2 ko'rsatmasidan 3 marta ko'p bo'lgan algoritmni bajardi. Agar INKOR holati vuz bermagan bo'lsa, Baga nechta bargga surildi va qavsi vo'nalishda?

III bob. PROTSEDURALAR – YANGI KO'RSATMALAR

Biz Ijrochini ba'zi yangi ko'rsatmalarga qanday o'rgatishni ko'rsatamiz. Yangi ko'rsatmalarni esa eskilaridan tuzamiz.

Algoritmik tafakkurning asosiy qismlaridan biri – **yozish** san'atidir. Bu so'z orqali biz ko'rsatmalarning murakkab tizimini sodda va tushunarli ko'rinishda yozish san'atini tushunamiz. Yozishni soddalashtirish uchun turli tuzilmalar xizmat qiladi. Bunday tuzilmalar juda ko'p algoritmlash tillariga xosdir. Ular quyidagi imkoniyatlarni beradi:

- eski ko'rsatmalar asosida yangilarini hosil qilish;
- ba'zi ko'rsatmalarni bir necha marta takrorlash;
- ba'zi shartlarni tekshirish va tekshirish natijasiga asosan ish ko'rish.

Endi bu ro'yxatni birinchi elementi bilan tanishamiz.

Har qanday algoritmi yangi ko'rsatmaga aylantirish mumkin. Buning uchun unga oddiy narsa – **nomni** qo'shib qo'yish yetarli. Haqiqatan, agar biz ko'rsatmani bajarmoqchi bo'lsak, u holda uni qandaydir usulda chaqirishimiz, ya'ni unga nomi bo'yicha murojaat qilishimiz kerak. Yangi ko'rsatmalarni eski ko'rsatmalardan farqlash uchun ularni **protseduralar** deb ataymiz. Demak,

protsedura = nomlangan algoritm

Qo'llanmada protsedurani nomlash uchun quyidagi kelishuvdan foydalanamiz:

PROT <yangi ko'rsatma protsedura nomi>

BOSHLANISH

<protsedura vazifasini aniqlovchi algoritm>

TAMOM

Bu sxemani bir necha misollarda namoyish etamiz.

1. Biz bo'ri, echki va karamni daryodan o'tkazish algoritmini yozib qo'rganmiz. Bu algoritm uchun nom o'ylab topamiz,

masalan, o'tkaz bek (bek – bu bo'ri, echki, karam). End protsedurani yozish qiyin emas:

PROT o'tkaz bek

BOSHLANISH

echkini o'tkaz

suzib o't

bo'rini o'tkaz

echkini o'tkaz

karamni o'tkaz

suzib o't

echkini o'tkaz

TAMOM

Bu ish bajarilgandan keyin biz faqat **o'tkaz bek** ko'rsatmasini bajarsak bo'ldi, bu esa **BOSHLANISH** va **TAMOM** so'zlari orasida yozilgan barcha yettita sodda ko'rsatmani bajarilishiga olib keladi.

3.1-masha

Quvidagi algoritm bajarildi:

o'tkaz bek

o'tkaz bek

o'tkaz bek

Bu yerda qanday hodisa yuz berdi?

Javob. Dehqon bo'ri, echki va karamni daryoning chap qirg'og'idan o'ng qirg'og'iga olib o'tdi. Shundan keyin darrov ularni daryoning o'ng qirg'og'idan chap qirg'og'iga olib o'tdi. Dehqon bu ishni tugatib, yana ularni daryoning chap qirg'og'idan o'ng qirg'og'iga olib o'tdi.

2. Ikkita sodda ko'rsatmali Chigirtkani qaraymiz. Soddalik va aniqlik uchun Chigirtkani ko'rsatmalari **oldinga** n va **orqaga** m bo'lganda **Chigirtka(n ; m)** deb belgilab olamiz.

Quyidagicha ko'rsatmali Chigirtkani qaraymiz

oldinga 7

orqaga 5

ya'ni, Chigirtka(7;5) ni qaraymiz. Uning uchun yangi **bir qadam oldinga** ko'rsatmasini quramiz:

PROT bir qadam oldinga

BOSHLANISH

oldinga 7

oldinga 7

oldinga 7

orqaga 5

orqaga 5

orqaga 5

orqaga 5

TAMOM

Tekshirib ko'rishingiz mumkin, barchasi to'g'ri!

Bu yangi ko'rsatma asosida 0 nuqtadan-3 nuqtaga o'tish juda oson:

bir qadam oldinga

bir qadam oldinga

bir qadam oldinga

Yoki 0 nuqtadan 10 nuqtaga:

oldinga 7

bir qadam oldinga

bir qadam oldinga

bir qadam oldinga

Ko'rib turganingizdek, algoritmda protsedurani Ijrochining oddiy ko'rsatmasi sifatida chaqirish mumkin ekan.

Protsedura boshqa protsedura hosil qilish uchun ham ishlatilishi mumkin:

PROT oldinga 16

BOSHLANISH

oldinga 7

bir qadam oldinga

oldinga 7

bir qadam oldinga

TAMOM

Yangi **oldinga 16** nomli protseduradan boshqa protsedura va ko'rsatmalar kabi foydalanish mumkin.

Sintaksis qoidalari:

- **PROT, BOSHLANISH** va **TAMOM** so'zlari bosh harflarda yoziladi.
- Bu so'zlar turli qatorda qat'iy kelish tartibida yozilishi shart.
- Protsedura nomi **PROT** so'zidan keyin shu qatorda yoziladi.
- Protsedura matni **BOSHLANISH** va **TAMOM** so'zlari orasida ozgina siljatilgan holda yoziladi.
- Protsedura nomida kichik harflar va raqamlar qo'llanilishi mumkin. Bosh harflarni qo'llash mumkin emas.

1-sharh

Sintaksis qoidalari barcha algoritmlash tillarida bor. Ular bir biridan Pascal, Basic, Logo tillaridagi kabi anchagina farqlanishi mumkin. Biz bu yerda o'z tilimiz qoidalarini yoritmoqdamiz.

2-sharh

*Protsedurani yoza turib unga nom o'ylab topamiz. Nomi tanlashda biz erkinmiz. Bu erkinlikdan Aql bilan foydalanish maqsadga muvofiq. Mana, masalan, Chigirtka protsedurasini **bir qadam oldinga** deb nomladik. Uni anjir yoki bo'lmasa, t21mal54 deb nomlashimiz mumkin edi. Sintaksis qoidalari buni ta'qiqlamaydi. lekin, tasavvur qilingki, bir necha oy o'tib siz t21mal54 ko'rsatmasiga duch keldingiz.*

Bu ko'rsatma nima ish qilishini tushunishingiz uchun qancha vaqtning ketadi? Siz tuzgan bu ko'rsatmani boshqa odam ko'rayotgan bo'lsa-chi? Shuning uchun quyidagi maslahatga amal qiling: protseduraga siz berayotgan nom protseduraning maqsadini qisqacha aks ettirsin.

3.1-masala

a) Chigirtka (7;5) ni bir birlik chapga suruvchi **bir qadam orqaga** nomli protsedura tuzing. Uni ishlashi to'g'riligini tekshiring.

b) 0 nuqtadan 6 nuqtaga o'tkazadigan algoritm tuzing.

d) Quyidagi algoritm ishlashi natijasini aniqlang:

bir qadam oldinga

bir qadam orqaga

3. Uchta A , B , C idishli Suvchini **Suvchi(A ; B ; C)** kabi belgilat olamiz. Sunday masalani qaraymiz: Suvchi A va B idishlar yordamida C idishda k litr suvni yig'ishni algoritmini tuzing.

Albatta, bu masalada k ning qiymati C idishning hajmidan oshib ketmasligi shart. Xayolga birinchi shunday g'oya keladi: Suvchi A va B idishlar yordamida 1 litr suvni (agar mumkin bo'lsa) o'lchab oladi va C idishga soladi. Bu ishni k marta takrorlaydi. Yomon emas, agar 1 litr suvni o'lchab olish algoritmini yangi ko'rsatma — protsedura ko'rinishida yozilsa.

3.2-masala

Suvchi(5;8;16) uchun uchinchi idishda 14 litr suvni yig'ish algoritmini tuzing.

Yechim. Avvalgi bobda 5 litrli va 8 litrli idishlar yordamida 4 litrni o'lchab olish algoritmini yozgan edik. Shu algoritimga mos jadvalini e'tiborga olib va yechimidan foydalanib quyidagi protsedurani tashkil etishimiz mumkin:

PROT 7 litr

BOSHLANISH

A ni to'ldir

A dan B ga quy

A ni to'ldir

A dan B ga quy

B ni bo'shat

A dan C ga quy

A ni to'ldir

A dan C ga quy

TAMOM

Endi asosiy algoritm mos ravishda quyidagicha bo'ladi:

7 litr

7 litr

3.3-masha

Suvchi (5;8;16) uchun uchinchi idishda 14 litr suvni yig'ishning vuqoridagidan farqli algoritmini tuzing.

3.3-masala

Suvchi (5; 8; 16) uchun uchinchi idishda 9 litr suvni yig'ish algoritmini tuzing.

Nazorat savollari va to'pshiriqlar

- 1. Protsedura nima?*
- 2. Protsedura tuzayotganda aav'si so'zlarning bo'lishi shart?*
- 3. Protseduralarni tuzish sintaksis ooidalarini sanab bering.*
- 4. Iirochi nima uchun protsedurani balara oladi?*
- 5. Oshiruvchi uchun 2 ga ko'davtir ko'rsatmasini almashtiruvchi protsedura tuzing.*

Qo'shimcha masalalar

Oshiruvchi uchun

O-3.1. Oshiruvchi ish boshlashidan avval ekranda 0 vonib turgandi. Oshiruvchi 32 sonini hosil ailishi asosiy algoritm 2 ta satrdan iborat bo'ladiagan **osh16** nomli protsedura tuzing.

O-3.2. Oshiruvchi ish boshlashidan avval ekranda 0 vonib turgandi. Oshiruvchi uchun avvalgi masalada tuzilgan osh16 nomli protsedura

vordamida 64 sonini hosil qilish protsedurasini tuzing. osh16 nom protsedura vordamida qanday sonlarni hosil qilish mumkin?

O-3.3. Oshiruvchi ish boshlashidan avval ekranda 0 vonib turgand Oshiruvchi uchun quyidagi protsedura tuzilgan:

PROT aa

BOSHLANISH

1 ni o'sh

2 ga ko'pavtir

1 ni o'sh

TAMOM

Asosiy dasturda aa protsedurasi 7 marta vozilgan bo'lsa, ekranda qanday son hosil bo'ladi?

O-3.4. Oshiruvchi ish boshlashidan avval ekranda 5 vonib turgandi Oshiruvchi uchun quyidagi protsedura tuzilgan:

PROT aa

BOSHLANISH

1 ni o'sh

2 ga ko'pavtir

1 ni o'sh

TAMOM

Asosiy dasturda aa protsedurasi 3 marta vozilgan bo'lsa, ekranda qanday son hosil bo'ladi?

O-3.5. Oshiruvchi ish boshlashidan avval ekranda 0 vonib turgandi. Oshiruvchi **1 ni o'sh** va **2 ga ko'pavtir** ko'rsatmalarini 3 martadan bajaragan. Bu ko'rsatmalarni shunday tartibda vozingki, eng katta son hosil bo'lsin.

a) shu katta sonni hosil qilish uchun asosiy dastur 2 ta satrdan iborat bo'ladigan, 3 ta satrdan iborat protsedura vozish mumkinmi?

b) agar mumkin bo'lsa, protsedura va asosiy dasturni vozing.

Chigirtka uchun

Ch-3.1. Chigirtka (4;3) ni 1 oadam orqaga sillituvchi protsedura tuzing.

Ch-3.2. Chigirtka (4;3) ni 2 oadam orqaga sillituvchi protsedura tuzing.

Ch-3.3. Chigirtka (4;3) ni 1 oadam oldinga sillituvchi protsedura tuzing.

Ch-3.4. Chigirtka (4;3) ni 2 oadam oldinga sillituvchi protsedura tuzing.

Ch-3.5. Chigirtka (7;4) ni 0 tartib raqamli kvadratdan -15 tartib raqamli kvadratga o'tkazuvchi asosiv algoritm 3 ta satrdan iborat bo'ladi^{gan} protsedura tuzing.

Ch-3.6. Chigirtka (2;5) ni 0 tartib raqamli kvadratdan -17 tartib raqamli kvadratga o'tkazuvchi asosiv algoritm 3 ta satrdan iborat bo'ladi^{gan} protsedura tuzing.

Ch-3.7. Chigirtka (7;5) ni -3 tartib raqamli kvadratdan 15 tartib raqamli kvadratga o'tkazuvchi asosiv algoritm 2 ta satrdan iborat bo'ladi^{gan} protsedura tuzing.

Ch-3.8. Chigirtka (7;5) ni -1 tartib raqamli kvadratdan 15 tartib raqamli kvadratga o'tkazuvchi asosiv algoritm 2 ta satrdan iborat bo'ladi^{gan} protsedura tuzing.

Ch-3.9. Chigirtka (3;2) tartib raqami 0 kvadratda tur^{gan} edi. U ikki marta bajar^{gan} quvida^{gi} protseduradan ikkita ko'rsatma o'chib ketdi:

PROT aar

BOSHLANISH

oldinga 3

...

...

orqaga 2

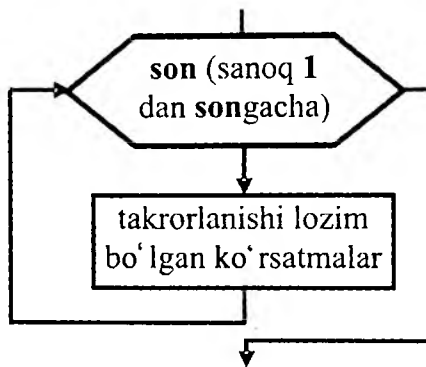
oldinga 3

TAMOM

Agar u 15 tartib raqamli kvadratga o't^{gan} bo'lsa, o'chib ket^{gan} ko'rsatmalarni vozing.

Undagi <son> o'rniga biror-bir aniq son kiritamiz, masalan, 30 yoki 77 yoki 189. Tuzilmaning mazmuni shundan iboratki, **TAKRORLANSIN 30 MARTA** va **TAMOM** so'zlari orasiga yozilgan barcha ko'rsatmalar sanoq 1 dan boshlab to 30 bo'lguncha takrorlanishi shart. Mana shunday! Bu tuzilmada ko'rsatmalar bittadan ortiq takroriy bajarilmoqda, bu kabi tuzilmalar **sikl** deb ataladi.

Buni qarangi, blok-sxemada bu tuzilma ham ko'zda tutilgan ekan:



4.1-rasm.

Endi 60 ta askar haqidagi masalaning yechimini yozish uchun avvalgisiga qaraganda ancha kam vaqt va qog'oz kerak bo'ladi:

TAKRORLANSIN 60 MARTA

ikkita bola

bola

askar

bola

TAMOM

4.1-masha

Bu algoritmnı blok-sxema ko'rinishida yozing.

Agar biror hazilkash xayoliga sizdan 60 ta emas, 600 ta askar uchun algoritm yozib berishni so'rash kelib qolsa, endi sizni qo'r-qita olmaydi. Siz faqat **TAKRORLANSIN 60 MARTA** satrini **TAKRORLANSIN 600 MARTA** satriga almashtirasiz, xolos, tamomvassalom!

TAKRORLANSIN tuzilmasi oxirida **TAMOM** so'zi nima uchun kerak

Quyidagi tuzilmaga e'tibor beramiz.

TAKRORLANSIN <son> MARTA

<takrorlanishi lozim bo'lgan ko'rsatmalar>

TAMOM

Sizga oxirgi satrda yozilgan **TAMOM** so'zining keragi yo'qdek tuyulishi mumkin. Lekin bunday emas. Murakkab algoritmlarda ba'zi ko'rsatmalar ketma-ketligini takrorlash kerak bo'ladi, boshqalarini esa yo'q. Ularni qanday farqlash mumkin?

Mana, masalan, bizning ro'yxatda yuqorida keltirilgan uch ko'rsatmaga qo'shimcha ravishda to'rtinchi ko'rsatma ham bo'lsin:

g'alaba raqsi

Qiyin ishlari tugagandan so'ng bolalar g'alaba raqsiga tushgilari kelib qolishi mumkin. Keling, oddiygina qilib yozib qo'ya qolamiz:

TAKRORLANSIN 60 MARTA

ikkita bola

bola

askar

bola

g'alaba raqsi

Endi tushunib bo'lmay qoldi, bolalar necha marta raqsga tushishlari kerak — 1 martami yoki 60 marta (ya'ni har qayiqni qaytarib olishganidan keyin). Agar boshqacha yozsak-chi:

TAKRORLANSIN 60 MARTA

ikkita bola

bola

askar

bola

TAMOM

g'alaba raqsi

Bu holda hamma narsa joyiga tushdi. **TAKRORLANSIN 60 MARTA** va **TAMOM** so'zlari orasiga yozilgan barcha ko'rsatmalar 60 marta takrorlanadi, boshqalari esa yo'q, ya'ni **g'alaba raqsi** ko'rsatmasi faqat bir marta bajariladi.

Sintaksis qoidalari:

- **TAMOM** so'zini **TAKRORLANSIN** so'zidan pastda uning to'g'risidan yoziladi, ular orasidagi boshqa ko'rsatmalar esa o'ngroqqa siljitib yoziladi. Bu esa faqat yozilgan algoritمنى o'qish oson bo'lishi uchunidir.
- **TAKRORLANSIN, MARTA** va **TAMOM** so'zlari bosh harflarda, boshqa barcha ko'rsatmalar kichik harflarda yoziladi.

Yangi tuzilmani qo'llash: Chigirtka

Kiritilgan tuzilma yordamida juda ko'p masalalar yechimini o'ta sodda ko'rinishda yozishimiz mumkin. Ikkita ko'rsatmali Chigirtkadan boshlaymiz (II bobga qarang):

oldinga 3

orqaga 2

4.1-masala

Chigirtka 0 nuqtadan 360 nuqtaga olib boradigan algoritm tuzing.

Javob:

TAKRORLANSIN 120 MARTA

oldinga 3

TAMOM

Keyingi masala bir oz qiyinroq.

4.2-masala

Chigirtka 0 nuqtada turibdi. Uni 1 dan 100 gacha bo'lgan nuqtalarda bir martadan bo'lishga majbur qiling.

Yo'llanma. Shu kabi masala avval uchragandi – u Chigirtka 1 dan 5 gacha bo'lgan nuqtalarda bir martadan bo'lishi kerak edi. Masala quyidagicha yechilardi:

oldinga 3

orqaga 2

oldinga 3

orqaga 2

oldinga 3

Bu algoritmni bajarilishi natijasida Chigirtkamiz 5 nuqtada keldi va u 0 dan 5 gacha bo'lgan kesmadan chiqib ketmadi. Agar shu ko'rsatmalar ketma-ketligini yana takrorlasak, Chigirtka 6, 7, 8, 9 va 10 nuqtalarda faqat bir martadan bo'ladi (tekshirib ko'ring) va 10 nuqtaga kelib to'xtaydi. Endi yechim o'z-o'zidan ma'lum.

Yechim. Birinchi algoritmimizni protseduraga aylantiramiz:

PROT beshta o't

BOSHLANISH

oldinga 3

orqaga 2

oldinga 3

orqaga 2

oldinga 3

TAMOM

U holda asosiy algoritm juda sodda ko'rinishga keladi:

TAKRORLANSIN 20 MARTA

beshta o't

TAMOM

Endi 2.9-masalaga qaytamiz. U ikkita ko'rsatmali Chigirtka haqida edi:

oldinga 7

orqaga 5

Chigirtka to'g'ri chiziqning ixtiyoriy nuqtasiga bora oladimi?
Javob: ha.

Bunga ishonch hosil qilish uchun III bobdagi **bir qadam oldinga** protsedurasini esga olamiz. Bu protsedurani necha marta xohlasak, shuncha marta ketma-ket takrorlab 2, 3, 4, ... nuqtalarga bora olamiz. Shuning uchun, agar, masalan, 72 nuqtaga borishimiz kerak bo'lsa, biz quyidagicha algoritmni yozamiz:

TAKRORLANSIN 72 MARTA

bir qadam oldinga

TAMOM

Bu algoritmning e'tiborli tomoni shundaki, u shu kabi har qanday masalani yechishning umumiy usulini beradi. Agar, masalan, 72 nuqtasi o'rniga 216 kerak bo'lsa, u holda algoritmdagi 72 o'rniga 216 yozamiz, shu bilan masala hal. Bunday algoritmlarni **umumiy** deb atashadi.

Bu algoritmning kamchiligi ham bor — u juda ko'p marta sakrashni talab etadi.

Bu algoritmni Chigirtkani 72 nuqtaga o'tkazuvchi quyidagi algoritm bilan taqqoslab ko'ring:

TAKRORLANSIN 11 MARTA

oldinga 7

TAMOM

orqaga 5

4.2-masho

Ikkala algoritmdagi sakrashlar sonini hisoblang va taqqoslang. $\frac{5}{2}$

4.3-masala

10q

Chigirtkani 216 nuqtasiga o'tkazuvchi sakrashlar soni artib kam bo'lishini ta'minlovchi algoritm tuzing. Oxirgi

Yangi tuzilmani qo'llash: Oshiruvchi

Avvalo II bobdagi bizga ma'lum bo'lgan ikkita algoritmnı boshqacha yozib olamiz. Ularning ikkalasi ham 0 sonidan 17 sonini hosil qiladi. Birinchi algoritm:

TAKRORLANSIN 17 MARTA

1 ni qo'sh

TAMOM

Ikkinchi algoritm:

1 ni qo'sh

TAKRORLANSIN 4 MARTA

2 ga ko'paytir

TAMOM

1 ni qo'sh

Birinchi algoritm juda sodda ko'rinadi. Bunga qo'shimcha u yana umumiy: 17 soni o'rniga ixtiyoriy sonni yozish mumkin va algoritm ishlashi natijasida shu sonni hosil qilamiz. Lekin u samarali emas, chunki natijaga erishish uchun birinchi algoritmga 17 qadam, ikkinchisiga esa faqatgina 6 qadam kerak bo'ladi.

4.3-masha

0 dan 1024 sonini hosil qiluvchi ikki algoritmnı taqqoslaymiz.

TAKRORLANSIN 1024 MARTA

1 ni qo'sh

TAMOM

1 ni qo'sh

TAKRORLANSIN 10 MARTA

2 ga ko'paytir

TAMOM

1. Ikkinchi algoritm ishlashi natijasida 1024 hosil bo'lishini tekshiring.

2. Ikkala algoritmdagi qadamlar sonini hisoblang.

Endi algoritmnı o'zingiz bajarishingiz kerak, deb faraz qiling: sizning oldingizda ekranida son aks etadigan avtomatik qurilma hamda 1 va 2 raqamli ikkita tugma bor. 1 tugmasining bosilishi ekrandagi sonni bittaga oshiradi; 2 tugmasining bosilishi ekrandagi sonni 2 ga ko'paytiradi. Avval ekranda 0 soni bor, siz 1024 sonini hosil qilmoqchisiz. Siz qaysi algoritmnı tanlagan bo'ladingiz: birinchisini yoki ikkinchisini?

4.4-masala

0 sonidan 729 sonini 30 qadamdan oshirmasdan hosil qiling.

Yangi tuzilmani qo'llash: Baga

Ijrochi Baqa uchun soddalik va aniqlik maqsadida belgilashlar kiritamiz, ya'ni ko'rsatmalari **oldinga d** , **oldinga k** , **orqaga m** va **orqaga h** bo'lganda **Baqa(d ; k ; m ; h)** deb belgilab olamiz. II bobda ko'rilgan Baqa(1; 2; 1; 2) uchun ko'rsatmalar 4 ta edi:

oldinga 1

oldinga 2

orqaga 1

orqaga 2

4.5-masala

Baqa n ta bargli nilufarning 1 tartib raqamli bargiga tushdi. U barcha pashshalarni yeb n tartib raqamli barg ustiga borish algoritmini tuzing.

Javob: Yechim juda sodda:

TAKRORLANSIN n MARTA

oldinga 1

TAMOM

4.6-masala

Baqa toq sondagi n ta bargli nilufarning 1 tartib raqamli bargiga tushdi. U barcha pashshalarni yeb 2 tartib raqamli barg ustiga borish algoritmini tuzing.

Yechim. Masala shartida Baqaning boshlang'ich nuqtasi va n soni toq ekanligi berilgan. Demak, toq tartib raqamli barglar ustida sakrab oxirgi nuqta n ga borish uchun **oldinga 2** ko'rsatmasini kerakli marta qo'llaymiz. Mayli-ku, lekin necha marta? Bu juda oson: son = $(n-1):2$ marta ($n-1$ soni juft ekanligini tekshirib ko'ring). Endi algoritmini yozish mumkin:

TAKRORLANSIN son MARTA

oldinga 2

TAMOM

Baqa n tartib raqamli barg ustida ko'p qololmaydi, shuning uchun masalani tezroq hal etishimiz kerak. Baqa endi **orqaga 2** ko'rsatmasini qo'llay olmaydi, chunki u sakrab o'tib ketgach, toq tartib raqamli barglar suv ostida ko'rinmay qoldi. Lekin juft tartib raqamli barglar orasida bitta barg sig'adigan masofa qoladi. Oxirgi

n tartib raqamli bargdan $(n-1)$ tartib raqamli barg ustiga o'tish uchun **orqaga 1** ko'rsatmasini qo'llaymiz. Endi Baqa yana ikkitalab sakrashga majbur, ya'ni **orqaga 2** ko'rsatmasini **son-1** marta qo'llaymiz (to'g'riligini tekshirib ko'ring), demak, to'liq algoritm quyidagicha bo'ladi:

TAKRORLANSIN son MARTA

oldinga 2

TAMOM

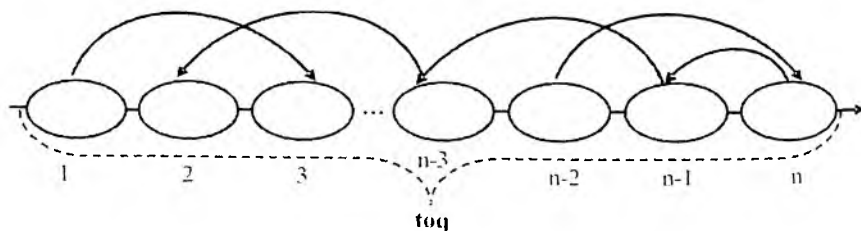
orqaga 1

TAKRORLANSIN son-1 MARTA

orqaga 2

TAMOM

Baqani bu algoritmni bajarishidagi sakrashlari quyidagi 4.2-rasmda strelkalar bilan ko'rsatilgan.



4.2-rasm.

4.4-masha

Baqa toa sonidagi n ta bargli nilufarning 2 tartib raqamli bargiga tushdi. U barcha pashshalarni veb 1 tartib raqamli barg ustiga borish algoritmini tuzing.

Yo'llanma. Agar Baqa 2 tartib raqamli bargdan keyin 3 tartib raqamli bargga o'tsa, hech qachon 1 tartib raqamli bargga qaytolmaydi. Shuning uchun, juft tartib raqamli barglar orqali o'ng tomonga borish lozim.

Xulosa. Baqa ikkita yonma-yon turgan barglarga sakrab, o'ng tomonga o'tgan bo'lsa, u hech qachon bu barglarni chap tomoniga o'ta olmaydi va aksincha, chap tomonga o'tgan bo'lsa, u hech qachon bu barglarni o'ng tomoniga o'ta olmaydi.

4.7-masala

Baqa juft sonidagi n ta bargli nilufarning 1 tartib raqamli bargiga tushdi. U barcha pashshalarni yeb, 2 tartib raqamli barg ustiga olib boradigan algoritm tuzing.

Yo‘llanma. Avvalgi masalalar yechimidan foydalaning.

4.8-masala

Baqa toq sondagi n ta bargli nilufarning 1 tartib raqamli bargiga tushdi. U barcha pashshalarni yeb, juft songa teng b tartib raqamli barg ustiga olib boradigan algoritmini tuzing.

Yo‘llanma. Baqa oldinga 1 ko‘rsatmasini qo‘llab ($b-1$) tartib raqamli barg ustiga boradi. Keyin esa 4.6-masala yechimidan foydalaniladi.

4.9-masala

Baqa juft sondagi n ta bargli nilufarning 1 tartib raqamli bargiga tushdi. U barcha pashshalarni yeb juft songa teng b tartib raqamli barg ustiga olib boradigan algoritm tuzing.

4.10-masala

Baqa juft sondagi n ta bargli nilufarning juft son bo‘lgan a tartib raqamli bargiga tushdi. U a tartib raqamli bargdan chapdagi barcha pashshalarni yeb, $a+1$ tartib raqamli barg ustiga olib boradigan algoritm tuzing.

4.11-masala

Baqa juft sondagi n ta bargli nilufarning juft son bo‘lgan a tartib raqamli bargiga tushdi. U a tartib raqamli bargdan chapdagi barcha pashshalarni yeb juft son bo‘lgan b tartib raqamli barg ustiga olib borish algoritmini tuzing.

Endi bizda suhbatlashish uchun 3 ta mavzu bor:

- Nima uchun **TAKRORLANSIN**, **MARTA** va **TAMOM** so‘zlarini bosh harflarda yozamiz?
- Algoritm sifatini qanday tekshirish mumkin: sodda yoki murakkab, samarali yoki yo‘q va hokazo?
- Oshiruvchi uchun samarali algoritmlarni qanday yozish mumkin?

Bularni navbati bilan qarab chiqamiz.

Nima uchun bosh harflar

Hozircha kam sondagi ijrochilar bilan tanishdik. Keyingi boblarda ularning soni ko‘payadi. Albatta, qo‘llanmada uchra-maydigan juda ko‘p sondagi Ijrochilarni o‘ylab topish mumkin.

Har bir Ijrochi uchun o'zining ko'rsatmalar (yoki qoidalar) tizimi bor. Lekin quyidagi takrorlash tuzilmasi

TAKRORLANSIN <son> MARTA

<takrorlanishi lozim bo'lgan ko'rsatmalar>

TAMOM

barchasi uchun bir xil. U qaysi Ijrochi bilan ish ko'rayotganimizga bog'liq emas va hammasiga mos kelaveradi. Tuzilma **universaldir** – har qanday Ijrochi uchun algoritm tuzishga imkon beradi. Bu tuzilma algoritmlashda asosiylaridan biridir.

Yuzlab va minglab Ijrochilar bor. Lekin **TAKRORLANSIN – MARTA** ko'rinishidagi universal tuzilmalar juda kam va bu algoritmlashning asos xossasidir. Asosiy tuzilmalar bor-yo'g'i uchta. Ba'zan qulaylik uchun bir nechta qo'shimcha tuzilmalar qo'llab ham turiladi. Agar siz qo'llanmaning asosiy maqsadi – bu tuzilmalardan foydalanish san'atini o'rgatish deb hisoblasangiz, deyarli xato qilmaysiz. Algoritmik tafakkur va algoritmlashning kaliti – bu universal tuzilmalarni erkin qo'llay bilishdir.

Doimo, universal tuzilmalarni tashqi ko'rinishidan Ijrochilar ko'rsatmasidan farqlash uchun bosh harflar bilan yozamiz.

Samaradorlik va murakkablik

Biz turli algoritmlarning samaradorligini muhokama qilib o'tdik. Algoritmni bajarilish qadami – bu Ijrochi tomonidan bitta ko'rsatmaning bajarilishidir. Bir masalani hal etuvchi ikkita algoritmdan kam qadam talab qilinayotgani samaraliroqdir. Samaradorlik o'lchovi – bu bor-yo'g'i qadamlar sonidir.

Lekin chuqurroq e'tibor berib qarasak, bu ta'rifdagi mujmal tomonlarni aniqlaymiz. Ba'zan avval uchragan algoritmlardagidan ko'ra vaziyat murakkabroq bo'ladi. Hozircha bu masala chuqurlashib o'tirmaymiz va chuqurroq bilim to'plamaguncha uni muhokamasini keyinga qoldiramiz.

Algoritmlar murakkabligi bilan ham farqlanishi mumkin. Algoritmning murakkabligini uning matnidagi satrlar soni bilan o'lchaymiz. Shu bilan birga quyidagi ikki satrni bir tuzilmaning ikki qismi bo'lgani uchun bittaga hisoblaymiz:

TAKRORLANSIN <son> MARTA

TAMOM

Mana, masalan, quyidagi algoritmda:

1 ni qo'sh

TAKRORLANSIN 6 MARTA

2 ga ko'paytir

1 ni qo'sh

TAMOM

faqat 4 ta satr:

1 ni qo'sh

TAKRORLANSIN 6 MARTA

TAMOM

2 ga ko'paytir

1 ni qo'sh

bor. Bu uning murakkabligi 4 ekanligini bildiradi.

Shuni aytib o'tish joizki, II bobdagi barcha algoritmlarning barchasini murakkabligi va samaradorligi o'zaro tengdir. Masalan, bo'ri, echki va karamni daryodan o'tkazish algoritmi ham 7 satrdan iborat ham u 7 qadamda bajariladi. Bu kabilar shu bobning barcha algoritmlari uchun ham o'rinli. Bizda shu vaqtgacha algoritm matnidagi satrlarini qisqartirish uchun vosita yo'q edi. Bizdan algoritm ko'rsatmasini bajarish talab etilgan bo'lsa, biz ko'rsatmani algoritm matniga joylashtirishga majbur edik. Masalan, Oshiruvchi tomonidan 17 sonini hosil qilish algoritmiga qarang: 17 marta bajarilish algoritm matnining 17 satriga aylandi.

Endi esa bizni kerakli vositamiz bor: bu **TAKRORLANSIN - MARTA** tuzilmasi. Shuning uchun Oshiruvchi tomonidan 17 sonini hosil qilish algoritmi 3 satrdan iborat bo'ladi (eslatamiz: tuzilma 1 ta satr deb hisoblanadi):

1 ni qo'sh

TAKRORLANSIN 16 MARTA

1 ni qo'sh

TAMOM

Endi bu algoritmning samaradorligi 17 ga, murakkabligi esa 17 emas, 3 ga teng. Askarlar va qayiq masalasi algoritmidagi 60 ta askarni daryodan o'tkazish uchun 240 qadam bajariladi, algoritm matni esa 5 satrdan iborat. Bu algoritmning samaradorligi 240 ga, murakkabligi esa 5 ga teng.

Baqa uchun tuzilgan 4.6 - masalani algoritmidagi qadamlar sonini hisoblaymiz:

$$\text{son} + 1 + \text{son} - 1 = (n - 1):2 + (n - 1):2 = n - 1.$$

Demak, har qanday n toq son uchun algoritmni samaradorligi $n - 1$ ga teng ekan. Algoritmning murakkabligi esa n toq son nechaga teng bo'lishidan qat'iy nazar, 5 ga teng bo'ladi!

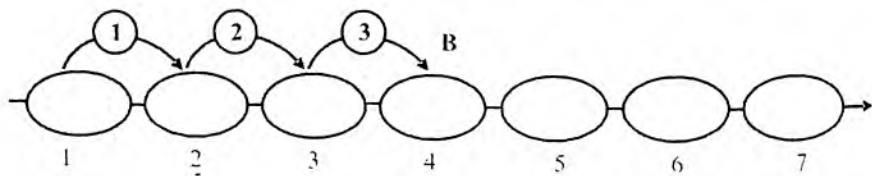
4.5-masho

Baqa uchun tuzilgan 4.7-masalani algoritmini samaradorligi va murakkabligini hisoblang.

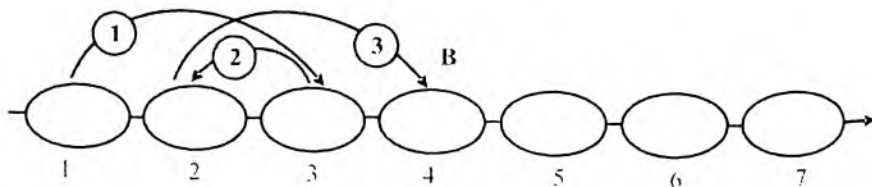
Hisoblab chiqqan bo'lsangiz, natijasi havron qoldirgandir?

Xulosa. Baqa masalasiga oid algoritmlarning samaradorligi faqat n sonining qiymatiga bog'liq. Chunki masala shartida Baqa har bir bargdagi pashshani yeb chiqishi talab qilinadi. U holda barglar soni n ta ekanligi va Baqa biror bargning ustida turgandan keyin harakat boshlanganligidan qadamlar soni doimo $n - 1$ ta bo'lishi kelib chiqadi.

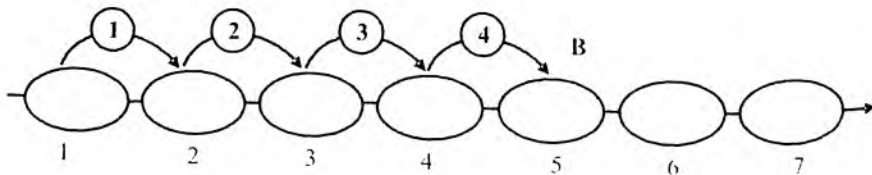
Haqiqatan, masalan, agar **1 tartib** raqamli bargdan **4 tartib** raqamli bargga o'tish kerak bo'lsa, u holda barcha imkoniyatlarni 4.3–4.4-rasmlarda, agar **1 tartib** raqamli bargdan **5 tartib** raqamli bargga o'tish kerak bo'lsa, u holda barcha imkoniyatlarni 4.5–4.7-rasmlardan ko'rishimiz mumkin.



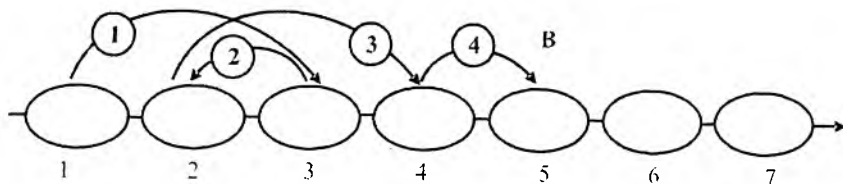
4.3-rasm.



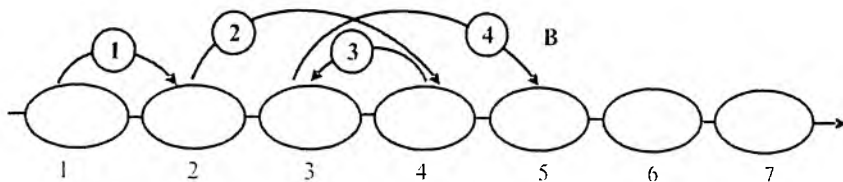
4.4-rasm.



4.5-rasm.



4.6-rasm.



4.7-rasm.

Va nihoyat, yana bir izoh. Samaradorlik va murakkablik talabi ko'pincha bir-biri bilan ziddiyatga kirishadi. Bu mutlaqo tabiiy. Axir, mashina sotib olayotgan bo'lsangiz, eng chiroyli va qulay mashinaning eng arzon bo'lishiga umid qilmaysiz. Algoritmashda ham shunday. Agar sizga juda samarador algoritm kerak bo'lsa, bu algoritm boshqalariga nisbatan ancha murakkabroq bo'lishi ehtimol. Amaliyotda esa oqilona murosaga kelishga to'g'ri keladi.

Samarador Oshiruvchi

Samarador algoritmlarni tuzish bizdan jiddiy aqliy tirishqoqlikni talab etadi. Bir xususiy holni ko'rib chiqamiz: Oshiruvchi tomonidan 0 sonidan 729 sonini hosil qilish masalasi.

Birinchi qadamda, ravshanki, 1 ni qo'shish kerak — 0 ni ko'paytirish befoyda. Endi 1 soni hosil bo'ldi. Ikkinchi qadamda 2 soniga ko'paytirish ham 1 sonini qo'shish ham mumkin — natija bir xil (ya'ni, 2) bo'ladi. Aniqlik uchun ko'paytirishni tanlaymiz.

Endi ekranda qaysi son yozilgan bo'lishidan qat'iy nazar, 1 ni qo'shishdan ko'ra 2 ga ko'paytirish 729 soniga yaqinlashishni tezlashtiradi.

Bir qaraganda, iloji boricha **2 ga ko'paytir** dan foydalanish kerakdek tuyuladi. Tanlangan usul quyidagi sonlar ketma-ketligiga olib keladi: $0 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16 \rightarrow 32 \rightarrow 64 \rightarrow 128 \rightarrow 256$

→ 512. Agar yana bir marta 2 ga ko'paytirsak, marradan o'tib ketamiz. Shuning uchun biz endi **1 ni qo'sh** ko'rsatmasini 217 marta ($729 - 512 = 217$) bajarishimiz kerak bo'ladi. Juda ham samarador emas, to'g'rimi?

Biror-bir tasodifan tanlangan oraliq qadamda **1 ni qo'shdan** foydalanishga harakat qilamiz.

Birinchi urinish: 1 ni 4 soniga qo'shamiz (shundan keyin 2 ga ko'paytirdan foydalanamiz): $0 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 10 \rightarrow 20 \rightarrow 40 \rightarrow 80 \rightarrow 160 \rightarrow 320 \rightarrow 640$. Bu holda 1 ni qo'sh ko'rsatmasini 89 marta ($729 - 640 = 89$) bajarishimiz kerak bo'ladi. Ancha yengillika erishdik!

Ikkinchi urinish: 1 ni 2 soniga qo'shamiz (shundan keyin 2 ga ko'paytir dan foydalanamiz): $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 12 \rightarrow 24 \rightarrow 48 \rightarrow 96 \rightarrow 192 \rightarrow 384$. Bu holda 1 ni qo'sh ko'rsatmasini 345 marta ($729 - 384 = 345$) bajarishimiz kerak bo'ladi. Endi ahvolimiz yomonlashdi-ku!

Bundan tushunarliki, 1 ni qo'sh ko'rsatmasidan o'ziga xos, ya'ni «eng yaxshi» qadamda foydalanishimiz kerak. Lekin uni qanday topish mumkin?

Quyidagi fikr juda ko'p o'yinlarda yordamga keladi: oxiridan boshlash. Keling, yangi Ijrochini hosil qilamiz va uni **Kamaytiruvchi** deb ataymiz. Uning ko'rsatmalari quyidagicha:

1 ni ayir

2 ga bo'l

Kamaytiruvchi 1 ni ayir ko'rsatmasi berilganda ekrandagi sondan birni ayiradi. Ekrandagi son 0 ga teng bo'lganda bu ko'rsatma **Inkor** holatiga olib keladi. Kamaytiruvchi **2 ga bo'l** ko'rsatmasi berilganda ekrandagi sonni teng ikkiga bo'ladi. Agar ekrandagi son toq bo'lsa, bu ko'rsatma **Inkor** holatiga olib keladi.

Endi 729 sonidan 0 sonini hosil qilamiz. Oshiruvchini **2 ga ko'paytir** ko'rsatmasi **1 ni qo'sh** ko'rsatmasiga nisbatan natijaga yaqinlashishni tezlashtirsa, Kamaytiruvchini **2 ga bo'l** ko'rsatmasi **1 ni ayir** ko'rsatmasiga nisbatan ham jarayonni tezlashtiradi. Shuning uchun ham imkon bo'lganda (Kamaytiruvchi uchun) har gal ikkiga bo'lishni xohlaymiz. Lekin endi ekrandagi sonning o'zi 2 ga bo'lish mumkinligini ko'rsatib beradi. Ya'ni, agar son juft bo'lsa, u holda uni 2 ga bo'lamiz, agar son toq bo'lsa, bu holda faqat undan 1 ni ayirishimiz mumkin (natijada yana juft son hosil bo'ladi). Kamaytiruvchi uchun algoritmnining boshlanishi quyidagicha:

Ko'rsatma	Son
	729
1 ni ayir	728
2 ga bo'l	364
2 ga bo'l	182
....	

4.6-masha

Algoritmni mustaqil tugating.

Endi Kamaytiruvchi uchun algoritm tayyor bo'lgach, uni Oshiruvchi uchun o'zgartiramiz.

Buning uchun algoritmni quyidan yuqoriga yozib boramiz, faqat **1 ni ayir** o'rniga **1 ni qo'sh** ko'rsatmasini, **2 ga bo'l** o'rniga **2 ga ko'paytir** ko'rsatmasini yozamiz. Natijada quyidagi ikkita algoritm hosil bo'ladi, biri Kamaytiruvchi uchun ikkinchisi Oshiruvchi uchun.

Qadamlar soni	Kamaytiruvchi ko'rsatmalari	Son	Oshiruvchi ko'rsatmalari	Son
0		729		0
1	1 ni ayir	728	1 ni qo'sh	1
2	2 ga bo'l	364	2 ga ko'paytir	2
3	2 ga bo'l	182	2 ga ko'paytir	4
4	2 ga bo'l	91	1 ni qo'sh	5
5	1 ni ayir	90	2 ga ko'paytir	10
6	2 ga bo'l	45	1 ni qo'sh	11
7	1 ni ayir	44	2 ga ko'paytir	22
8	2 ga bo'l	22	2 ga ko'paytir	44
9	2 ga bo'l	11	1 ni qo'sh	45
10	1 ni ayir	10	2 ga ko'paytir	90
11	2 ga bo'l	5	1 ni qo'sh	91
12	1 ni ayir	4	2 ga ko'paytir	182
13	2 ga bo'l	2	2 ga ko'paytir	364
14	2 ga bo'l	1	2 ga ko'paytir	728
15	1 ni ayir	0	1 ni qo'sh	729
Bu aloritmlarda qadamlar soni bor-yo'a'i 15 ga teng!				

4.12-masala

1000 dan kichik har qanday sonni 0 sonidan ko'pi bilan 20 qadamda hosil qilish mumkinligini ko'rsating.

Yo'llanma. Agar qadamlar soni 20 tadan ko'p bo'lsa, ikkilantirishlar soni nechta? – ... dan kam emas.

Oshiruvchi uchun eng «qulay» sonlar – ikkining darajalaridir, ya'ni 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024,... Ularni qo'shish amalisiz faqat ikkilantirish orqali hosil qilishimiz mumkin. Oshiruvchi uchun eng «noqulay» sonlar – ikkining darajasidan bittaga kam sonlar, ya'ni 1, 3, 7, 15, 31, 63, 127, 255, 511, 1023... Bu sonlarni hosil qilish uchun har ikkilantirishdan so'ng bittaga kamaytirib hosil qilish mumkin (tekshirib ko'ring!).

4.13-masala

Oshiruvchi uchun 0 sonidan 14 ta qadamda hosil qilib bo'lmaydigan sonni toping.

Javob: 255.

4.14-masala

Oshiruvchi uchun 0 sonidan 15 ta qadamda hosil qilib bo'lmaydigan sonni toping.

Javob: $256+127 = 383$.

4.15-masala

Kamaytiruvchi uchun 729 sonidan 17 sonini hosil qiluvchi eng samarador algoritmi tuzing.

Oshiruvchi uchun algoritmi yaxshilash

Berilgan a sonidan b sonini hosil qilish algoritmi tuzilgan bo'lsin. Agar $a < b$ bo'lsa, u holda quyidagilar o'rinli ekanini isbotlang:

- agar algoritmda birinchi a ta ko'rsatma **1 ni qo'sh** bo'lsa, u holda ularni **2 ga ko'paytir** ko'rsatmasi bilan almashtirish mumkin;
- agar algoritmda ketma-ket kelgan quyidagi ko'rsatmalar guruhi bor bo'lsa

2 ga ko'paytir

1 ni qo'sh

1 ni qo'sh

u holda ular o'rniga quyidagi ko'rsatmalar guruhini almashtirib, algoritmi yaxshilash mumkin:

1 ni qo'sh
2 ga ko'paytir

Masalan, 3 dan 10 ni hosil qilish algoritmini yozamiz:

1 ni qo'sh	}	→	{	2 ga ko'paytir	}	{	1 ni qo'sh	1 ni qo'sh
1 ni qo'sh	}			1 ni qo'sh	}		2 ga ko'paytir	1 ni qo'sh
1 ni qo'sh	}			1 ni qo'sh	}		1 ni qo'sh	2 ga ko'paytir
1 ni qo'sh				1 ni qo'sh			1 ni qo'sh	
1 ni qo'sh				1 ni qo'sh			1 ni qo'sh	
1 ni qo'sh				1 ni qo'sh			1 ni qo'sh	
1 ni qo'sh				1 ni qo'sh			1 ni qo'sh	

Oxirgi algoritmni yaxshilab bo'lmaydi.

Nazorat savollari va to'ntirishlar

1. Sikl deganda nima tushuniladi?
2. Samaradorlik deganda nimani tushunasiz? Misollar bilan izohlang.
3. Murakkablik deganda nimani tushunasiz? Misollar bilan izohlang.
4. Siklik tuzilishli algoritmlarni sintaksis qoidalari qanday?
5. Qanday algoritmlar universal deyiladi?
6. Qanday tuzilmalar universal deyiladi?
7. Ibrochi Kamaytiruvchi ko'rsatmalar ro'yxati haqida so'zlab bering.
8. Ibrochi Kamaytiruvchi uchun sodda amallarni yozib chiqing.
9. Ibrochi Kamaytiruvchi uchun bor bo'lsa kamida bitta INKOR holatni yozing.
10. Havotingizdan sikllarga misollar keltiring.
11. Bobda keltirilgan barcha mashalarni bajarib.

Qo'shimcha masalalar

M-1. Darvo qirg'og'iga 60 ta askar kelishdi. Ular darvoning nariqi qirg'og'iga o'tishlari kerak. Qirg'og' vaqinida to'rtta o'z qavida suzib yuribdi. Qavida shunchalik kichkinaki, faqat ikkita askarni ko'tarishi mumkin. Darvo odami juda tez bo'lganidan qavida ag'darilib ketmasligi uchun qavida ikkita dan kam o'z bo'lishi mumkin emas. Qanday qilib askarlar darvodan o'tib olishadi va qaviani qizlarga qavtarib berishadi? Ibrochi uchun ko'rsatmalar ro'yxatini ishlab chiqing va algoritm tuzing.

Chiqirtka (3;2) bo'ya ko'rsatmasini bajarishni o'rgandi.

Ch-1. Chiqirtka boshlang'ich holati — 0 tartib raqamli kvadrat. Quvidagi algoritmlar bajarilgandan kevin natijada qanday farq bo'ladi?

1- algoritm
oldinga 3
bo'ya

2- algoritm
bo'ya
oldinga 3

Ch-2. Chigirtka -1 tartib raqamli kvadratdan harakatni boshladi va quyidagi algoritmi bajaradi:

oldinga 3

bo'ya

oldinga 3

oraqaga 2

bo'ya

oraqaga 2

U qaysi kvadratlarni bo'yaagan?

Ch-3. Chigirtka 0 tartib raqamli kvadratda turibdi. U 0 dan katta 500 dan kichik har uchinchi kvadratni bo'vash algoritmini tuzing.

Ch-4. Chigirtka 0 tartib raqamli kvadratda turibdi. U 0 dan katta 500 dan kichik har to'rtinchi kvadratni bo'vash algoritmini tuzing.

Ch-5. Chigirtka 0 tartib raqamli kvadratda turibdi va uni bo'vadi. U -500 dan katta va 500 dan kichik har to'rtinchi kvadratni bo'vash algoritmini tuzing.

V bob. MANTIQ ELEMENTLARI VA SHARTLAR

Mantiq elementlari

Har bir dasturlash tilida biror-bir shartni tekshirish imkoni bor. Shartni tekshirish deganda biror narsani da'vo qilib, uni rost yolg'onligini aniqlashni tushunamiz. Masalan, biz son juft deb da'vo qildik deylik, agar son haqiqatan ham juft bo'lsa, u holda da'vomiz **ROST**, aks holda da'vomiz **YOLG'ON** bo'ladi. Demak, shartni tekshirish degani juft son da'voyimiz o'rinlimi, degan savol berish kabi ekan. Faqat bu savolga **HA** deb yoki **YO'Q** deb javob beriladi. Shu sababli blok-sxemada, avval ko'rganingizdek, quyidagi ko'rinishdagi blok qaraladi:



Agar da'vomiz shartning qisqa ko'rinishi ekanligini hisobga olsak, u holda oddiy ko'rinishi qanday bo'ladi? Mana bunday: **tekshirilayotgan shart o'rinli**. Endi bu da'vo rost yoki yolg'onligi haqida fikr yuritish mumkin bo'ladi.

Hayotimizda juda ko'p bu kabi da'volarni aytamiz. Masalan: «Bugun havo issiq», «0 soni juft», «Qishda qor yog'adi», «Men algoritmikani yaxshi ko'raman» va hokazo. Bu barcha da'volar **ROST** bo'lavermaydi, boshqacha aytganda, sharoitga bog'liq bo'ladi. Haqiqatan, birinchi gapni qishning barcha kunlarida, uchinchi gapni esa Afrika qishida, to'rtinchi gapni esa hammamiz ham ayta olmaymiz. Faqatgina ikkinchi gap doimo **ROST** bo'ladi.

E'tibor qilgan bo'lsangiz, bu gaplarning har biri darak gapdir, ya'ni bizga axborot bermoqda. Bu gaplarning **ROST** yoki **YOLG'ON** bo'lishi mazmunan berilayotgan axborotmi **ROST** yoki **YOLG'ON** bo'lishi bilan o'zaro bir qiymatli bog'liq.

Ko'pincha darak gaplarda biz ikki va undan ortiq axborot beramiz yoki biror narsani inkor etamiz. Masalan, «Bugun havo bulut va yomg'ir yog'yapti», «2 juft va tub son», «Kurashda men yutaman yoki sen yutasan», «Men imtihondan o'taman yoki

imtihondan yiqilaman», «Oynani sindirgan men emas» va hokazi «va» hamda «yoki» bog'lovchili gaplarni alohida ajratish mumkinmi ya'ni alohida da'vo sifatida qarash mumkinmi? Agar ularning ma'nosini yo'qotishni istamasak, ajratishimiz mumkin emas. U holda bu kabi murakkab da'volar shart sifatida qanday qo'llaniladi? Bu savolga javob quyida keltirilgan.

Murakkab (birikkan) shartlarni yozish uchun juda qulay vosita - mantiqiy amallar o'ylab topilgan. Ular uchta:

VA, YOKI, EMAS

VA mantiqiy amali. Umumiy ko'rinishi: <shart 1> VA <shart 2> VA mantiqiy amali ikkita shartni biriktirib, bitta qo'shma shartga aylantiradi: **VA amali orqali hosil qilingan qo'shma shart rost bo'ladi, shunda va faqat shundaki, agar biriktirilayotgan ikkala shart ham rost bo'lsa.** Agar shartlardan biri yolg'on bo'lsa, u holda qo'shma shart ham yolg'on bo'ladi. Masalan, 2 sonini ham juftligi, ham tubligi sharti quyidagicha yoziladi: **2 juft VA 2 tub.**

Agar har bir shartni (da'voni yoki darak gapni) bosh lotin harflari bilan belgilab olsak, natijaning ta'rifga asosan mantiqiy qiymati qanday bo'lishini quyidagi jadval orqali ko'rishimiz mumkin.

A shart (da'vo)	B shart (da'vo)	A VA B qo'shma shart
ROST	ROST	ROST
ROST	YOLG'ON	YOLG'ON
YOLG'ON	ROST	YOLG'ON
YOLG'ON	YOLG'ON	YOLG'ON

YOKI mantiqiy amali. Umumiy ko'rinishi: <shart 1> YOKI <shart 2>. YOKI mantiqiy amali ikkita shartni biriktirib, bitta qo'shma shartga aylantiradi: **YOKI amali orqali hosil qilingan qo'shma shart rost bo'ladi, shunda va faqat shundaki, agar biriktirilayotgan ikkala shartdan hech bo'lmaganda bittasi rost bo'lsa.** Agar shartlarning ikkalasi yolg'on bo'lsa, u holda qo'shma shart ham yolg'on bo'ladi. Masalan, xonadagi o'quvchi chap devor yoki to'g'ridagi devor yoniga kelib qolganligi sharti quyidagicha yoziladi: **to'g'ri bo'sh emas, YOKI chap bo'sh emas.**

Belgilashlardan keyin bu mantiqiy amal uchun ham ta'rifga asosan jadval tuzamiz:

A shart (da'vo)	B shart (da'vo)	A YOKI B ao'shma shart
ROST	ROST	ROST
ROST	YOLG'ON	ROST
YOLG'ON	ROST	ROST
YOLG'ON	YOLG'ON	YOLG'ON

EMAS mantiqiy amali. Umumiy ko'rinishi: **EMAS <shart>**. EMAS mantiqiy amali berilgan shartga aks shartni hosil qiladi. **Aks shart rost bo'ladi, agar shart yolg'on bo'lsa va aksincha.**

Xafa bo'lmasin deb belgilashlardan keyin bu mantiqiy amal uchun ham ta'rifga asosan jadval tuzamiz:

A shart (da'vo)	EMAS A shart (da'vo)
ROST	YOLG'ON
YOLG'ON	ROST

Yuqorida keltirilgan jadvallarni ko'pincha **rostlik jadvali** deb atashadi.

Mantiqiy amallar o'z nomiga ega. VA amali **konyuksiya**, YOKI amali **dizyunksiya** va EMAS esa **inkor** amali deyiladi. Shu oxirgi gapga e'tibor qilsangiz, biz uchta da'vo yozdik, chunki vergul ham VA kabi qo'llanilgan.

Mantiqiy amallarni barcha Ijrochining har qanday shartlariga qo'llash mumkin.

Shuning uchun ular ham, TAKRORLANSIN – MARTA tuzilmasi kabi bosh harflarda yoziladi.

Murakkab mantiqiy birikmalarda qavs qo'llanilib, arifmetik amallardagi kabi avval ichki qavslar ichidagi Shart qiymati hisoblanadi.

Masalan, quyidagi qo'shma shart qiymati ROST (tekshirib ko'ring):

((EMAS 3 juft) VA 4 juft) VA 5 toq

Qiziqarli mantiqiy masalalar

Mantiqiy amallar mantiq ilmida ham algoritmik tafakkurni rivojlantirishda ham juda katta ahamiyatga ega. Masalan, quyidag masalani qaraylik.

5.1- masala.

Bir kishi aytdi «Men yolg'onchiman yoki qora sochliman». U kishi kimligini aniqlang.

Yechim. Masala shartidagi da'volar uchun belgilashlar kiritamiz

$D =$ «Men yolg'onchiman yoki qora sochliman»;

$A =$ «Men yolg'onchiman»; $B =$ «Qora sochliman»

U holda masala shartidagi da'voni shunday yoza olamiz: $D=A$ **YOKI** B . Bu amal uchun rostlik jadvali quyidagicha ko'rinishda bo'ladi

A	B	D=A YOKI B
ROST	ROST	ROST
ROST	YOLG'ON	ROST
YOLG'ON	ROST	ROST
YOLG'ON	YOLG'ON	YOLG'ON

Endi masala yechimini topish uchun quyidagicha mulohaza yuritamiz:

a) agar $A =$ ROST bo'lsa, u holda masala shartidagi da'voni aytgan kishi yolg'onchi bo'ladi va shuning uchun uning hamma gapi yolg'on. Demak, $D =$ YOLG'ON bo'lishi kerak. Lekin jadvaldan ko'rinadiki, $A =$ ROST bo'lganda $D=YOLG'ON$ bo'lolmaydi.

b) agar $A =$ YOLG'ON bo'lsa, u holda masala shartidagi da'voni aytgan kishi rostgo'y bo'ladi va tabiiyki, uning hamma gapi rost. Demak, $D =$ ROST bo'lishi kerak. Jadvaldan ko'rinadiki, bunday imkoniyat $A =$ YOLG'ON va $B=ROST$ bo'lsagina bor.

Javob: masala shartidagi da'voni aytgan kishi ROSTGO'Y va QORA SOCHLI ekan.

5.2-masala

Bir kishi aytdi: «Men yolg'onchiman va qora sochliman». U kishining kimligini aniqlang.

Yechim. Avvalgi masaladagi kabi shartdagi da'volar uchun belgilashlar kiritamiz:

D = «Men yolg'onchiman va qora sochliman»;

A = «Men yolg'onchiman»; B = «Qora sochliman»

Masala shartidagi da'voni shunday yoza olamiz: $D = A \vee B$.

Bu amal uchun rostlik jadvali quyidagicha ko'rinishda bo'ladi:

A	B	$D=A \vee B$
ROST	ROST	ROST
ROST	YOLG'ON	YOLG'ON
YOLG'ON	ROST	YOLG'ON
YOLG'ON	YOLG'ON	YOLG'ON

Masala yechimini topish uchun quyidagicha mulohaza yuritamiz:

a) agar $A = \text{YOLG'ON}$ bo'lsa, u holda masala shartidagi da'voni aytgan kishi rostgo'y bo'ladi va tabiiyki, uning hamma gapi rost. Demak, $D = \text{ROST}$ bo'lishi kerak. Lekin jadvaldan ko'rinadiki, $A = \text{YOLG'ON}$ bo'lganda $D = \text{ROST}$ bo'lolmaydi.

b) agar $A = \text{ROST}$ bo'lsa, u holda masala shartidagi da'voni aytgan kishi yolg'onchi bo'ladi va tabiiyki, uning hamma gapi yolg'on. Demak, $D = \text{YOLG'ON}$ bo'lishi kerak. Jadvaldan ko'rinadiki, bunday imkoniyat $A = \text{ROST}$ va $B = \text{YOLG'ON}$ bo'lsagina bor.

Javob: masala shartidagi da'voni aytgan kishi **YOLG'ONCHI** va **QORA SOCHLI EMAS** ekan.

5.3-masala

Uchta do'st futbol bo'yicha 2010 yilgi jahon chempionati natijalari haqida bahslashishardi.

«Mana ko'rasiz, Ispaniya chempion bo'lmaydi, Germaniya chempion bo'lishi aniq», dedi Abror.

«Yo'g'e, Ispaniya chempion bo'ladi, Argentina haqida gapirma ham bo'ladi, u chempion bo'lolmaydi», dedi Behzod.

«Germaniya chempionlikka yaqin ham kelmaydi, lekin Argentinada zo'r o'yinchilar bor», dedi Muzaffar.

Chempionat tugagandan keyin esa qarashsa, uch do'stdan ikkitasini ikkala gapi ham to'g'ri, uchinchisini ikkala gapi ham noto'g'ri ekan. Kim chempion bo'lgan?

Yechim. Ba'zi belgilashlarni kiritib olamiz:

A – Argentina chempion, G – Germaniya chempion, I – Ispaniya chempion.

Muzaffarning «Argentinada zo'r o'yinchilar bor» degan ga kim chempion bo'lishi haqida hech qanday ma'lumot bermay, shuning uchun keyingi mulohaza yuritishimizga qaralmaydi. H bir do'stning gapini belgilab olamiz:

Abror: (EMAS I) VA G; Behzod: I VA (EMAS A); Muzaff: EMAS G.

Do'stlarning ikkitasini ikkala gapi ham to'g'ri, uchinchi ikkala gapi ham noto'g'ri ekanligini hisobga olib, quyidagi da'voni hosil qilamiz:

$$D = ((EMAS I) VA G) VA (I VA (EMAS A)) VA (EMAS(EMAS G)) YOKI ((EMAS I) VA G) VA (EMAS (I VA (EMAS A))) VA (EMAS G) YOKI (EMAS ((EMAS I) VA G)) VA (I VA (EMAS A)) VA (EMAS G) = ROST$$

Agar yuqoridagi da'vo uchun erinmasdan rostlik jadvalini tuzi chiqsangiz, kim chempion bo'lganini bilib olasiz, balki bilarsiz ham (yechimimizni tekshirib ko'ring!).

Albatta, mantiq nazariyasida yuqoridagi D mantiqiy ifodan soddalashtirish uchun formulalar bor. Lekin ular qo'llanmad: ko'rib chiqilmaydi.

Takomillashtirilgan Kamaytiruvchi

Eslatib o'tishimiz kerakki, Kamaytiruvchining ikkitagina ko'rsatmasi bor:

1 ni ayir
2 ga bo'l

O'tgan bobda Kamaytiruvchi uchun samarador algoritmi tuzish uchun ekrandagi har bir sonni tekshirgan edik — son juftmi yoki toqmi. Tekshirish natijasiga ko'ra tegishli ko'rsatmani qo'llaganmiz — juft sonni 2 ga bo'lganmiz, toq sondan 1 ni ayirganmiz. Bu tekshirishlarni barchasini o'zimiz bajarganmiz, lekin bu ishni algoritimga topshirishni xohlar edik.

Juft degan shart kiritamiz. Bu shartni tekshirish — «Ekrandagi son juft, to'g'rimi» degan savol berish bilan bir xil. Bu savolga javoblar soni ikkita: **ROST** va **YOLG'ON**. **ROST** javobi shart bajarilganini, ya'ni ekrandagi son juftligini bildiradi. **YOLG'ON** ekrandagi son toqligini bildiradi.

Qo'llanmada shartli tuzilmaning 3 ta shaklini qo'llaymiz. Birinchisi va ulardan eng soddasi quyidagi ko'rinishga ega:

AGAR <shart>

U HOLDA

<ko'rsatmalar guruhi>

TAMOM

Masalan, namuna sifatida quyidagicha tuzilmani yozish mumkin:

AGAR juft

U HOLDA

2 ga bo'l

TAMOM

Agar juft degan shart rost bo'lsa, u holda ekrandagi son 2 ga bo'linadi. Agar yolg'on bo'lsa, hech qanday ko'rsatma bajarilmaydi.

Yana e'tibor qildingizmi, **AGAR, U HOLDA, TAMOM, ROST** va **YOLG'ON** so'zlarini bosh harflarda yozayapmiz? Ha, siz to'g'ri tushundingiz: bizga yana Ijrochiga bog'liq bo'lmagan universal tushunchalar uchradi. Kamaytiruvchi ayirish va bo'lish ko'rsatmalarini bajara oladi. Lekin **AGAR – U HOLDA** tuzilmalari esa ixtiyoriy Ijrochi uchun o'rinli va ularning barchasi uchun bir xil ma'noga ega. **ROST** va **YOLG'ON** – shartning universal rostlik qiymati bo'lib, barcha Ijrochilar uchun umumiy va ularning shartlariga bog'liq emas, lekin juftlik sharti esa faqat Kamaytiruvchiga xosdir.

AGAR tuzilmasining **ikkinchi** ko'rinishi quyidagicha:

AGAR <shart>

U HOLDA

<birinchi ko'rsatmalar guruhi>

AKS HOLDA

<ikkinchi ko'rsatmalar guruhi>

TAMOM

Bu holda Ijrochi, agar shart qiymati rost bo'lsa, u holda (**U HOLDA** va **AKS HOLDA** so'zlari orasidagi) birinchi guruh ko'rsatmalarni bajaradi, agar shart qiymati yolg'on bo'lsa, u holda (**AKS HOLDA** va **TAMOM** so'zlari orasidagi) ikkinchi guruh ko'rsatmalarni bajaradi.

Tuzilmaning ikkinchi shakli birinchisiga qaraganda, umumiyroqdir. Agar shart qiymati yolg'on bo'lganda hech narsa qilmoqchi bo'lmasak, **AKS HOLDA** dan keyingi guruhni bo'sh qoldirishimiz mumkin.

Bu holda yuqoridagi namuna quyidagicha yoziladi:

AGAR juft
U HOLDA
2 ga bo'l
AKS HOLDA
TAMOM

Endi Kamaytiruvchi uchun samarador harakatlar juda sodda yoziladi:

AGAR juft
U HOLDA
2 ga bo'l
AKS HOLDA
1 ni ayir
TAMOM

1-sharh

*0 – juft son. Shuning uchun, yuqorida yozilgan **AGAR – U HOLDA** – **AKS HOLDA** tuzilma 0 hosil bo'lsa, uni 2 ga bo'ladi natijada yana 0 hosil bo'ladi.*

Yangi tuzilma yordamida Kamaytiruvchi biz (ekrandagi sonning juftligini tekshirish) tekshirishimiz lozim bo'lgan shartni o'zi tekshirmoqda. Bu yangi tuzilmaning afzallik tomonlaridan biri, xolos.

Uning yanada muhim boshqa afzalligi ham bor. Avvalgi bobda har xil sonlar uchun turlicha algoritmlarni yozayotgan edik. Endi esa samaradorlikni saqlab qolgan holda universal algoritimga ega bo'ldik. Quyidagi algoritm 1000 dan kichik ixtiyoriy sonni 0 ga keltiradi:

TAKRORLANSIN 20 MARTA
AGAR juft
U HOLDA
2 ga bo'l
AKS HOLDA
1 ni ayir
TAMOM
TAMOM

Bizga birdaniga sodda, samarador va umumiy bo'lgan algoritmini hosil qilish uchun juda kamyob imkoniyat uchradi, bu esa juda qulay bo'lgan yangi **AGAR – U HOLDA – AKS HOLDA** tuzilmasining sharofati bilandir. Shu bilan yakunlash mumkin edi. Lekin Kamaytiruvchi algoritmini yaxshilashning ikki imkoniyati bor. Ularni ko'rib chiqamiz.

Birinchisi shundan iboratki, toq sondan 1 ni ayirganda biz albatta juft sonni hosil qilamiz. Shuning uchun, 1 ni ayirgandan keyin sonni juft yoki toqligini tekshirishning hojati yo'q. Son juft, shuning uchun o'ylab o'tirmasdan 2 ga bo'lishimiz mumkin va tekshirishni bo'lishdan keyin bajarsak ham bo'ladi. Bu mulohazalar quyidagi algoritmgga olib keladi:

AGAR juft

U HOLDA

2 ga bo'l

AKS HOLDA

1 ni ayir

2 ga bo'l

TAMOM

Algoritmning qo'shimcha afzalligi shundan iboratki, 1000 dan kichik har qanday son uchun bu ko'rsatmalar guruhini avvalgidek 20 marta emas, 10 marta bajarish yetarli (chunki ikkiga bo'lish har bir takrorlanishda bajarilmoqda):

TAKRORLANSIN 10 MARTA

AGAR juft

U HOLDA

2 ga bo'l

AKS HOLDA

1 ni ayir

2 ga bo'l

TAMOM

TAMOM

5.1-masho

Algoritmni to'liq vozimg va 100; 299; 35 sonlariga qo'llang.

Agar biz **AGAR – U HOLDA – AKS HOLDA** guruhi to'liq bajarilganda 35 dan boshlab barcha sonlarni vozib chiqsak, ushbu ketma-ketlikni hosil qilamiz:

$35 \rightarrow 17 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0.$

5.2-masho

Ketma-ketlikda nima uchun 11 ta had, 10 ta emas?

Yo'llanma. Sonlar orasida nechta strelka bor?

Endi ikkinchi imkoniyatni ko'rishga o'tamiz. Algoritmning kamchiligi ko'rinib turibdi: algoritmda 0 hosil bo'lgandan keyin ham to'xtamayapti. U **AGAR – U HOLDA – AKS HOLDA** guruhini roppa-rosa 10 marta bajarmoqda.

Bu to'siqni olib tashlash uchun yangi shart kiritishimiz zarur. **musbatlik.**

Eslatib o'tamiz: 0 musbat son emas. Endi bu shartga javobdan foydalanish usulini ko'rsatishimiz kerak. Shuning uchun takrorlash tuzilmasining yangi ko'rinishini kiritamiz. Yangi tuzilma takrorlanishlar soni oldindan belgilanmaydi. Bu tuzilmada ko'rsatmalar guruhi sharoitdan kelib chiqib, necha marta kerak bo'lsa shuncha marta bajariladi:

AGAR musbat

U HOLDA

AGAR juft

U HOLDA

2 ga bo'l

AKS HOLDA

1 ni ayir

2 ga bo'l

TAMOM

TAMOM

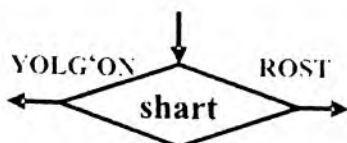
5.3-mashq

Algoritmning musbat, juft da'volarini shu da'volarga bir o'ymal mos keladigan da'volarga almashtiring.

Yo'llanma. Ishlatilmasa Inkor amali nega kerak?

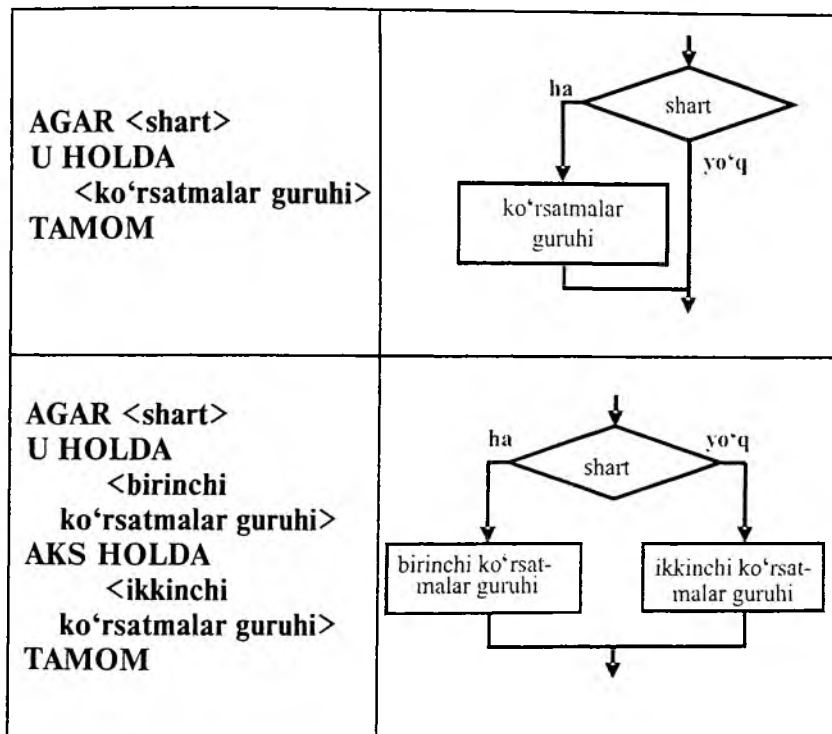
Tarmoqlanuvchi algoritmlar blok-sxemalari

Kompyuter belgilardan tuzilgan matnlarni tushunadi. Shuning sababli dasturchi kompyuter uchun dastur tuzayotganda uni biroz maxsus tilda (dasturlash tilida) yozadi. Insonga esa rasmlarni kuzatish osonroq. Shunga ko'ra dasturchi ish borishini kuzatish oson bo'lishi uchun, ko'pincha algoritmnı grafik ko'rinishida tasavvur qiladi. Biz ham o'zimizga oson bo'lishi uchun yuqoridagi tuzilmalarni blok-sxema ko'rinishida ifodalaymiz.



Shart tekshirish blokiga, ya'ni rombgaga, bir nechta strelka kelishi mumkin, lekin faqat ikkita strelka chiqadi. Agar romb ichidagi shart rost bo'lsa, u holda **ROST** (ha) strelka bo'ylab, aks holda **YOLG'ON** (yo'q) strelka bo'ylab yuramiz.

Tuzilmalarni algoritmik tilda hamda blok-sxema ko‘rinishida tasvirlaymiz, bunda ularning tarmoqlanishi yaqqol ko‘rinadi:



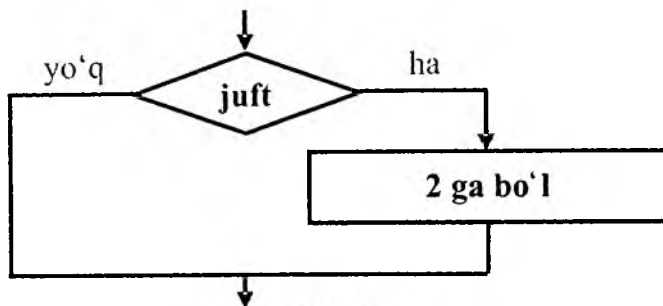
Har bir to‘g‘ri to‘rtburchakning ichida bir emas, bir nechta ko‘rsatma (ular ichida tuzilma ham uchrashi mumkin) borligiga e‘tibor bering. Bunday blok-sxemalar algoritmda ko‘rsatmalar soni ko‘p bo‘lganda juda qulaydir.

Jadvalda ko‘rganimizdek, **AKS HOLDA**dan keyingi guruh bo‘sh bo‘lishi mumkin, shuning uchun blok-sxemada aks ettirilmasligi mumkin (5.1-rasm). Masalan:

**AGAR juft
U HOLDA
2 ga bo‘l
TAMOM**

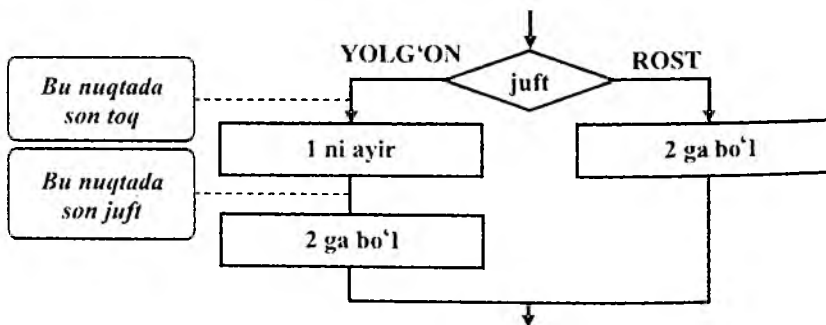
yoki

**AGAR juft
U HOLDA
2 ga bo‘l
AKS HOLDA
TAMOM**



5.1-rasm.

Quyida (5.2-rasm) samarador Kamaytiruvchi uchun blok-sxema keltirganmiz:



5.2-rasm.

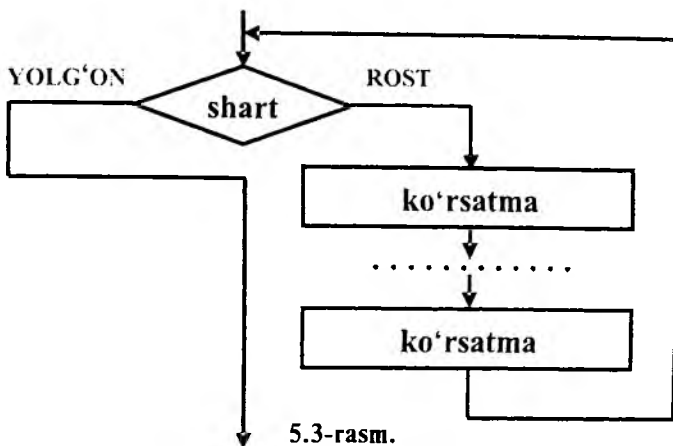
TOKI – BAJAR takrorlash tuzilmasi

Yangi **uchinchi** tuzilmaning ko'rinishi quyidagicha:

TOKI <shart> **BAJAR**
 <ko'rsatmalar guruhi>
TAMOM

Bu tuzilma qanday ishlashini ko'rib chiqamiz. Avval **TOKI** so'zidan keyingi javobi **ROST** yoki **YOLG'ON** chiqadigan savol beriladi. Agar javob **ROST** bo'lsa, **BAJAR** va **TAMOM** so'zlari orasidagi ko'rsatmalar guruhi bajariladi. Bajarish jarayoni tugagandan so'ng yana **TOKI** so'zidan keyin yozilgan savol beriladi. Agar javob **ROST** bo'lsa, yana (**BAJAR** va **TAMOM** so'zlari orasidagi) ko'rsatmalar guruhi bajariladi. Shundan keyin uchinchi, to'rtinchi va hokazo marta savolga qaytiladi. Bu takrorlanish jarayoni savolga javob **YOLG'ON** bo'lguncha davom etaveradi va javob **YOLG'ON** bo'lgandan keyingina to'xtaydi.

Yangi tuzilmaning blok-sxema ko'rinishi quyidagicha (5.3-rasm):



Kamaytiruvchi algoritmini bu ajoyib tuzilma yordamida quyidagicha osongina yozishimiz mumkin:

TOKI musbat BAJAR

AGAR juft

U HOLDA

2 ga bo'l

AKS HOLDA

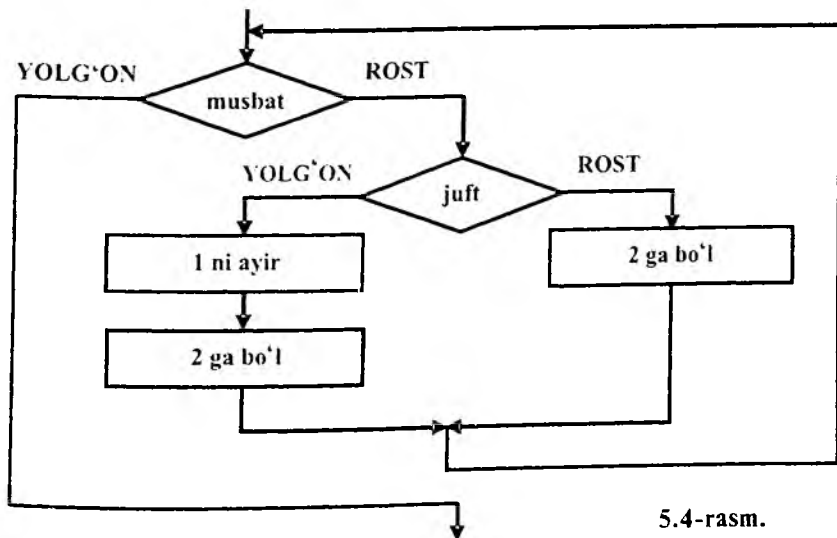
1 ni ayir

2 ga bo'l

TAMOM

TAMOM

Bu algoritmgga mos blok-sxema 5.4-rasmida keltirilgan:



5.4-masha

Bu algoritmni 13 soniga qo'llang. Ibrochining ko'rsatmalarni bajarish sonini (to'g'ri to'rtburchaklarğa tushishlar sonini) va shartlarni tekshirish sonini (romblarğa tushishlar sonini) hisoblang.

O'tgan boblarda tuzilganidek, «holatlar» jadvalini vozib chiqish mumkin. Ko'rsatmaning bajarilish natijasi ikki xil bo'lishi mumkin: birinchisi ekrandagi son, ikkinchisi shartlarni rostlik qiymati:

Ko'rsatma	Son	Shart	
		juft	musbat
	13	YOLG'ON	ROST
1 ni ayir	12	ROST	ROST
2 ga bo'l	6	ROST	ROST
2 ga bo'l	3	YOLG'ON	ROST
.....			

5.5-masha

Jadvalni mustaqil vakunlab qo'ying.

Endi ko'rsatmalarni bajarilish sonini hisoblash osonroq bo'lib qoldi.

5.4-masala

Kamaytiruvchi uchun yana bir algoritmni yozamiz:

TOKI musbat BAJAR

TOKI juft BAJAR

2 ga bo'l

TAMOM

1 ni ayir

TAMOM

- Bu algoritmga mos blok-sxemani chizing.
- Ikkala algoritmni 13; 1024; 1023 sonlari uchun qo'llang. Ikkala holda ham qadamlar sonini hisoblang.
- Nima uchun bu algoritmlar cheksiz vaqt bajarilmaydi?

5.5-masala

Quyida ikkita algoritm keltirilgan.

Birinchisi:

TOKI musbat BAJAR

1 ni ayir

TOKI juft BAJAR

2 ga bo'l

TAMOM

TAMOM

Ikkinchisi:

TOKI juft BAJAR

1 ni ayir

TAMOM

Bu algoritmlarning qanday kamchiligi bor?

TOKI — BAJAR tuzilmasiga ba'zi sharhlar

2-sharh

*O'z oldimizga bunday maqsad qo'yimagan bo'lsak-da, faqat 1000 dan kichik sonlar uchun emas, balki barcha natural sonlar uchun qo'llash mumkin bo'lgan algoritm tuzdik. Algoritmning o'zi nechta qadam kerak bo'lsa, shuncha bajaradi. Masalan, 1000000 soni uchun Kamaytiruvchi 29 ta ko'rsatma bajaradi. Agar son kattaroq bo'lsa, uning algoritmining bajarilishi ko'proq qadam talab qilishi mumkin. Bu tuzilmada biz endi **TAKRORLANSIN - MARTA** tuzilmasidagi kabi takrorlanishlar soni haqida o'ylashimiz shart emas. Endi takrorlanishlar uchun algoritmning o'zi javob beradi!*

3-sharh

***TOKI — BAJAR** tuzilmasining bajarilishi shart tekshirishdan boshlanadi. Birinchi savolga javob **YOLG'ON** bo'lsa-chi? Bu holda algoritm bajarilishi darrov to'xtatiladi va **BAJAR** va **TAMOM** orasidagi ko'rsatmalar bir marta ham bajarilmaydi. Masalan, 5.4-rasmdagi blok-sxemada boshlang'ich quymat 0 ga teng bo'lsa, u holda musbat sharti **YOLG'ON** bo'ladi, shunga ko'ra algoritm bajarilishi to'xtatiladi.*

4-sharh

*Agar shartga javob doimo **ROST** bo'lsa-yu, bir marta ham **YOLG'ON** bo'lmasa-chi? U holda algoritm ishi hech qachon to'xtamaydi. U cheksiz uzoq vaqt davom etadi. Masalan, quyidagi algoritmgga*

TOKI juft BAJAR

2 ga bo'l

TAMOM

boshlang'ich qiymat sifatida 0 sonini kiritamiz. 0 – juft son shuning uchun **juft** sharti **ROST**, 0 ni 2 ga bo'lsak, yana sonini hosil qilamiz. Ko'rsatmalarining bajarilishi yana boshida boshlanadi va cheksiz marta davom etadi. Dasturchilar bunda holatni **siklga tushib qoldi** deyishadi (yoki «cheksiz siklga tush qoldi»).

Albatta, siklga tushib qolishi yoqimli emas. Dasturchilar doim bu holatga tushib qolmaslik choralarini ko'rishlari shart. Shunga ham bu holat yuz bergan bo'lsa, u holda klaviaturadan bir yoki bir nechta klavishalar juftligi yordamida algoritm ishini to'xtatish mumkin.

5-sharh

Tajribasiz dasturchilar, ichki ko'rsatmalarni bajarayotgan kompyuter doimo **TOKI** shartini nazarda tutib turadi va tekshirib boradi, deb o'ylashadi.

Lekin bunday emas, kompyuter shartni barcha ichki ko'rsatmalarni bajarib bo'lgachgina tekshiradi. Buni quyidagi algoritm misolida ko'rsatamiz:

TOKI juft BAJAR

2 ga bo'l

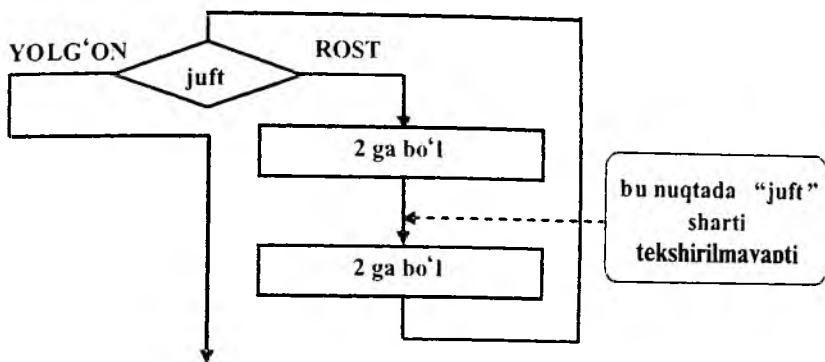
2 ga bo'l

TAMOM

Bu algoritmni 24 soniga qo'llashni jadval orqali ko'ramiz:

Ko'rsatma	Son	Shart	Natiija
		juft	
	24	ROST	
2 ga bo'l	12	ROST	
2 ga bo'l	6	ROST	
2 ga bo'l	3	YOLG'ON	
2 ga bo'l			INKOR

Algoritmida **INKOR** vaziyati yuzaga kelishi ikkinchi marta bo'lishdan avval biz **juft** shartini tekshirmayapmiz. Bu vaziyat blok sxemada quyidagicha aks etadi:



5.5-rasm.

6-sharh

TAKRORLANSIN – MARTA takrorlash ko'rsatmasini TOKI – BAJAR takrorlash ko'rsatmasi bilan osongina almashtirish mumkin.

Keling, 5.4-rasmdagi blok-sxemadagi qadamlar sonini taxminiy hisoblab ko'ramiz. Har gal biz blok-sxema orqali tepadan pastga o'tganimizda Kamaytiruvchi bitta yoki ikkita ko'rsatmani bajaradi. Bir marta o'tishda son, hech bo'lmaganda ikki marta kamayadi (ixtiyoriy yo'lda son ikkiga bo'linadi). Shuning uchun, biz agar blok-sxema orqali, masalan, 10 marta o'tsak, Ijrochi 10 tadan 20 tagacha qadam o'tadi, son esa hech bo'lmaganda $2^{10} = 1024$ marta kichrayadi.

Bir million taxminan 2^{20} ga teng (2^{20} dan kichikroq ham, tekshirib ko'ring: $2^{20} = 2^{10} \cdot 2^{10} = 1024 \cdot 1024 = ?$). Demak, bizga millionni 0 ga aylantirish uchun kamida $20 \cdot 1 = 20$ va ko'pi bilan $20 \cdot 2 = 40$ qadam kerak bo'ladi.

Bu baholash millionga yaqin har qanday son uchun o'rinli, aytaylik, 600000 dan 1000000 gacha sonlar uchun.

Nazorat savollari va to'vshiriolar

1. Algoritmikada o'andav shartlar bilan ish ko'riladi?
2. Mantioiv amallar haoida so'zlab berin \mathcal{E} .
3. Mantioiv amallar o'andav ketma-ketlikda bajariladi?
4. Biror mantioiv amalni rostlik iadvali haoida so'zlab berin \mathcal{E} .
5. Tarmoolanish tuzilmasinin \mathcal{E} oisaa ko'rinishi haoida so'zlab berin \mathcal{E} .
6. Tarmoolanish tuzilmasinin \mathcal{E} to'lio ko'rinishi haoida so'zlab berin \mathcal{E} .
7. Tarmoolanish tuzilmasi siklik tuzilmadan nimasi bilan faralanadi?

8. Inson havotidan tarqoolanish tuzilmasiga misollar keltiring.
9. TOKI – BAJAR tuzilmasiga misollar keltiring.

Qo'shimcha masalalar

1-mantiyiv masala. EMAS($6 > 5$) VA ($7+7 = 15$) YOKI ($63-21 =$ mantiyiv ifodaning mantiyivini aniqlang.

2-mantiyiv masala. A = ROST, B = YOLG'ON, C = ROST bo'lsa, A VA EMAS B YOKI C mantiyiv ifodaning mantiyivini aniqlang.

3-mantiyiv masala. $(A \text{ VA } B) \equiv ((\text{EMAS } A) \text{ YOKI } (\text{EMAS } B))$ mantiyiv o'rinli ekanligini rostlik jadvali yordamida isbotlang.

4-mantiyiv masala. $(A \text{ YOKI } B) \equiv ((\text{EMAS } A) \text{ VA } (\text{EMAS } B))$ mantiyiv o'rinli ekanligini rostlik jadvali yordamida isbotlang.

5-mantiyiv masala. Bir kishi aytdi: «Rost emaski men qoraman» to'g'ri yoki noto'g'ri? U kishi kim?

Mantiyiv masala-6. A kishi aytdi: «Hech bo'lmaganda birinchi qoraman», B kishi aytdi: «Hech bo'lmaganda birinchi yo'lg'onchimiz. Ular kimlar?

B-1. Barcha (1; 2; 1; 2) n ta bargli nilufarning a tartib raqam bargidan barcha pashshalarni va $a+1$ tartib raqamli bargiga kelitirish algoritmini tuzing.

B-2. Barcha (1; 2; 1; 2) n ta bargli nilufarning 1 tartib raqam bargidan barcha pashshalarni, va b tartib raqamli bargiga kelitirish algoritmini tuzing.

B-3. Barcha (1; 2; 1; 2) n ta bargli nilufarning a tartib raqamli bargida barcha pashshalarni va a dan o'ngdagi b tartib raqamli bargiga kelitirish algoritmini tuzing.

Robot qayerda yashaydi?

Robot teng o'lchamdagi kvadratlarga bo'lingan tekislikda yashaydi (6.1-rasm). U kvadratlarning birida joylashgan va ixtiyoriy qo'shni kvadratga o'tishi mumkin. Robot **qo'shni** deganda gorizontal yoki vertikal bo'yicha o'zi turgan kvadratga yopishgan kataklarni tushunadi. Shu bilan birga Robot o'zi turgan kvadratni bo'yashi ham mumkin. Rasmda Robot nuqta bilan, qo'shni kvadratlar *Q* harfi bilan ko'rsatilgan.

			Q				
		Q	•	Q			
			Q				

6.1-rasm.

Robot 5 ta ko'rsatmani bajaradi:

yuqoriga
quyiga
chapga
o'ngga
bo'ya

Bulardan **yuqoriga**, **quyiga**, **chapga** va **o'ngga** ko'rsatmalari Robotni mos yo'nalishlar bilan siljishga majbur qiladi. Lekin **bo'ya** ko'rsatmasida Robot harakatlanmaydi, faqat o'zi turgan kvadratni bo'yaydi. Agar kvadrat bo'yalgan bo'lsa u holda **bo'ya** ko'rsatmasida kvadratning rangi o'zgarmaydi (xuddi Robot hech qanday ish bajarmagandek bo'ladi).

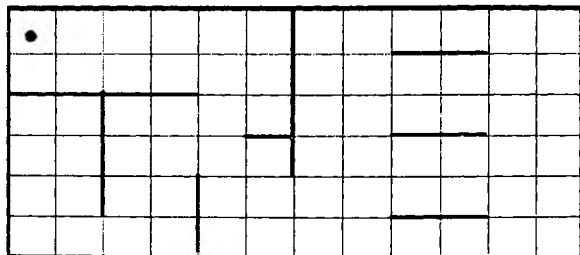
Devorlar

Robotning hayotida eng qizig'i, ba'zi kvadratlar orasida devor borligi (6.2-rasm). Odatda, Robot kvadratlardan hosil bo'lgan va har tomondan devorlar bilan o'ralgan to'g'ri to'rtburchak ichida

joylashgan bo'lad. Lekin shu to'g'ri to'rtburchak ichida ha devorlar bo'lishi mumkin.

Ba'zan devorlar murakkab shaklni hosil qiladi, bu shaklni **labiri** deb ataladi.

Robot devor ichidan o'ta olmaydi. Agar devor ichida o'tmoqchi bo'lsa, Robot «sochilib» ketadi. Yoki algoritmik til aytganda **INKOR** holatiga olib keladi.



6.2-rasm.

Robotni bunday halokatli holatlarga tushirmaslik uchun quyida to'rtta shartni tekshirishimiz zarur:

yuqori bo'sh

quyi bo'sh

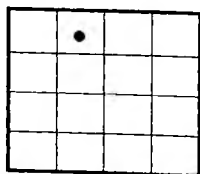
chap bo'sh

o'ng bo'sh

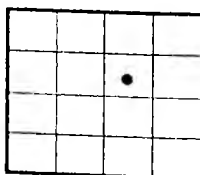
Bo'sh so'zi shu tomonda devor yo'qligini bildiradi.

Robot o'zi turgan katakning devorinigina aniqlay oladi. O'zi turgan kvadrat bilan devor orasida bitta kvadrat bo'lsa ham uzoqda bu devorni «ko'ra» olmaydi. U yonida turgan devorgagina «tegib ko'rishi» mumkin. Quyidagi 6.3-rasmda turli holatlarda **yuqori bo'sh** degan birgina shartning qiymatini ko'rish mumkin. Tushunarlikki **yuqori bo'sh** sharti (agar u **Rost** bo'lsa) Robot **yuqoriga** ko'rsatmasini sochilib ketmasdan bajara olishini bildiradi.

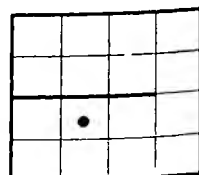
yuqori bo'sh



YOLG'ON



ROST



YOLG'ON

6.3-rasm.

Bu kabi mulohazalar **chap bo'sh** sharti va **chappa** ko'rsatmasi, va yana boshqa ko'rsatma va shart juftliklari uchun ham to'g'ri bo'ladi.

6.1-masha

Barcha ko'rsatma va shart juftliklari uchun rasmlarni chizib ko'ring.

Shartlar ro'yxatni tugatish uchun Robot biladigan oxirgi shartni keltiramiz:

bo'yalgan

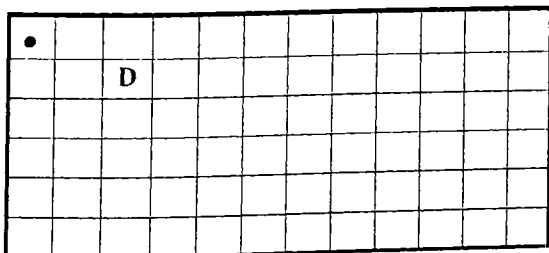
Bu shart Robot turgan kvadratni bo'yalgan yoki bo'yalmaganini tekshirish imkonini beradi. Agar kvadrat bo'yalgan bo'lsa, shart **ROST**, aks holda **YOLG'ON**.

Ko'rib turibsiz, Robotning ko'rsatmalari juda sodda. Lekin uni o'rab turgan muhit xilma-xil imkoniyatlarga boy. Robotning maydonida turli labirintlar, yo'laklar, har xil shakldagi xonalar va boshqa figuralar yordamida juda ko'p qiziqarli masalalar qo'ysih mumkin. Robotning mikrohayoti - algoritmik tafakkurni rivojlantirish uchun a'lo darajadagi mashq maydonidir.

Robotning imkoniyatlari bilan tanishuv

6.1-masala

Robotni turgan kvadratidan *D* harfi turgan kvadratga o'tkazuvchi Ot yurish nomli protsedurani yozing (6.4-rasm).



6.4-rasm.

Javob:

PROT ot yurish
BOSHLANISH
o'ngga

**o'ngga
quyiga**

TAMOM

Shubhasiz, bizning yechimimiz yagona emas, boshqa yechim ham bor.

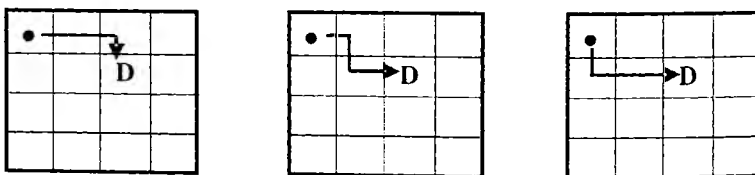
6.2-mashu

Bizning yechimimizdan farqli faqat uchta ko'rsatmadan iborat bo'lgan yana bitta otosedura vozining.

6.2-masala

Robotni turgan kvadratidan D harfi turgan kvadratga o'tka zuvchi 3 ta qadamdan iborat barcha algoritmlarni tuzing. Bunda algoritmlar soni nechta?

Yo'llanma. Bunday algoritmlar soni 3 ta. 6.5-rasmda shu algoritmlarga mos Robotning harakatlanish yo'li ko'rsatilgan



6.5-rasm.

6.3-mashu

6.1-masala yechimida keltirilgan algoritmgaga qavsi rasm mos kelishini aniqlang.

Javob. Birinchi.

6.3-masala

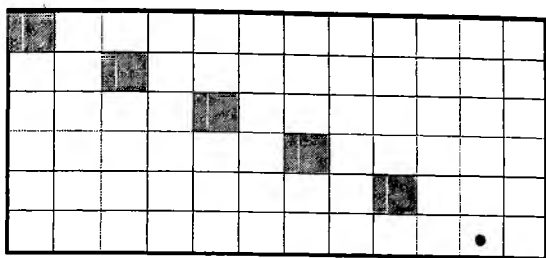
Robot 6×12 kvadratdan iborat devorlar bilan o'ralgan to'g'ri to'rtburchakli maydonning chap yuqori burchagida turibdi. Quyidagi algoritm natijasida hosil bo'lgan Robot maydonini chizing:

TAKRORLANSIN 5 MARTA

**bo'ya
ot yurish**

TAMOM

Javob. 6.6-rasmni qarang.



6.6-rasm.

6.4-masala

Oldingi masaladagi **bo'ya** ko'rsatmasi va **ot yurish** protsedurasi o'rinlarini almashtiramiz. Algoritm natijasida hosil bo'lgan Robot maydonini chizing:

TAKRORLANSIN 5 MARTA

ot yurish

bo'ya

TAMOM

6.5-masala

Yuqoridagi maydonda algoritmdagi takrorlanishlar soni 5 ni 6 bilan almashtiramiz. Algoritmni bajarishga urinishda nima ro'y berishini izohlang:

TAKRORLANSIN 6 MARTA

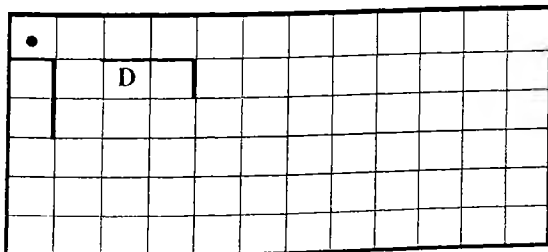
ot yurish

bo'ya

TAMOM

Javob. Algoritm **ot yurish** protsedurasi 5 marta muvaffaqiyatli bajariladi va oltinchi marta ishlash boshlanadi. Lekin Robot o'ngga bitta qadam qo'ya oladi. O'ngga yana bir qadam qo'yishga urinishda Robot devorga urilib sochilib ketadi.

Endi Robotning 6x12 maydonida devorlar quramiz (6.7-rasm).



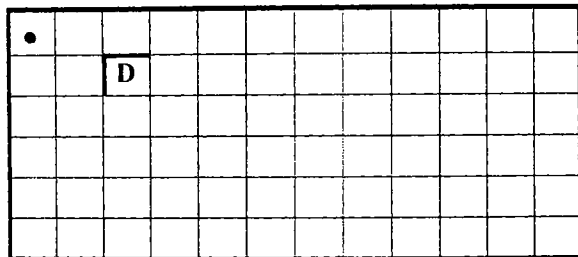
6.7-rasm.

6.6-masala

Robot D harfli kvadratga borishi kerak. 6.2-masalada yozilg algoritmlardan faqat bittasi bu masala yechimini beradi. S algoritmni toping.

Javob. Bu algoritm ikkinchi rasmga mos keladi.

Endi maydon ichidagi devorlar joyini o'zgartiramiz (6.8-rasm)



6.8-rasm.

Endi Robot 3 ta qadamda D harfli kvadratga bora olmaydi

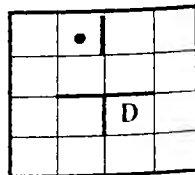
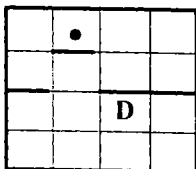
6.7-masala

Robotni 5 qadamda D harfli kvadratga olib keladigan algoritm yozing. Bu ishni bajara oladigan nechta bir-biridan farqli algoritm tuzish mumkin? Robotning yurish yo'lini chizing.

Yo'llanma. Bunday algoritmlar 4 ta.

6.8-masala

Quyidagi 3 ta rasm uchun (6.9-rasm) Robotni D harfli kvadratga olib keladigan algoritmlarni yozing.



6.9-rasm.

6.9-masala

Quyida keltirilgan algoritmlarda 5 marta **bo'ya** ko'rsatma qo'llanilgan:

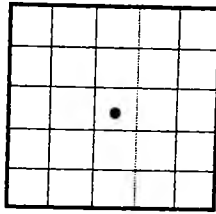
bo'ya
o'ngga
bo'ya

bo'ya
o'ngga
bo'ya

pastha
 bo'ya
 o'ngga
 bo'ya
 pastha
 bo'ya

pastha
 bo'ya
 chapga
 bo'ya
 yuqoriga
 bo'ya

Ikkala holda ham 6.10-rasmdagi maydonda bo'yalgan kvadratlar soni 5 ta bo'ladimi?

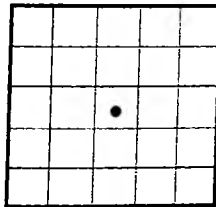


6.10-rasm.

Javob. Birinchi algoritmdagi Robot haqiqatan beshta kvadratni bo'yaydi. Ikkinchi algoritm faqat to'rtta katakni bo'yaydi, chunki Robot bitta kvadratni ikki marta bo'yaydi.

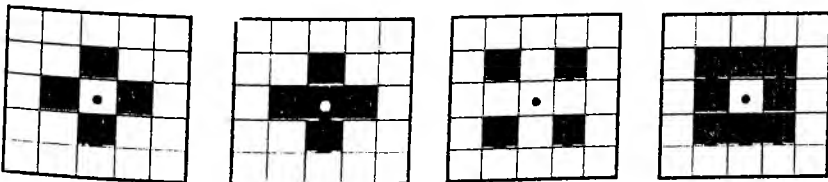
6.10-masala

Robotning dastlabki holati 6.11-rasmda ko'rsatilgan.



6.11-rasm.

Shunday algoritmlar tuzingki, ular bajarilgandan so'ng 6.11-rasmdan 6.12-rasmdagi naqshlar hosil bo'lsin.



6.12-rasm.

6.11-masala

O'zingiz bo'yalgan kvadratlardan iborat rasm o'ylang. Bu si-ning do'stingiz uchun vazifa bo'lsin, do'stingiz esa siz uchun vazifa tayyorlasin. Robot rasmlarni bo'yashi uchun algorit- tuzinglar.

Endi shumtakalar haqida 3 ta masalani e'tiboringizga havo- etamiz.

6.12-masala

Islom doskaga Robot uchun algoritm yozdi. Shumtaka Amon- yugurib kelib bitta ko'rsatmani o'chirib tashladi:

yuqoriga

o'ngga

???

quyiga

chapga

chapga

Agar Islomning algoritmi bajarilgandan keyin Robot o'zini- dastlabki joyiga qaytib kelishi ma'lum bo'lsa, Amon qanday ko'- rsatmani o'chirgan?

Yechim. Bu ko'rsatma **o'ngga**. Chunki Robot yana o'zini- dastlabki joyiga qaytib kelishi uchun **o'ngga** va **chapga** ko'rsatmalari- soni bir xil bo'lishi kerak. Xuddi shunday, **yuqoriga** va **quyig** ko'rsatmalari soni bir xil bo'lishi kerak.

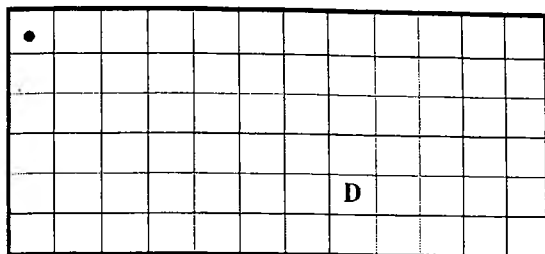
6.13-masala

Jamshid Robotni o'zining dastlabki joyiga qaytarib keltiradigan- algoritm tuzdi. Shumtaka Amon yugurib kelib bitta ko'rsatman- o'chirib tashladi. Ammo hosil bo'lgan algoritm bajarilgandan keyin- Robot baribir o'zining dastlabki joyiga qaytib keldi. Amon qanday- ko'rsatmani o'chirgan?

Javob. Bu ko'rsatma **bo'ya**.

6.14-masala

Sunnatilla Robotni *D* harfli kvadratga olib keladigan algoritm- tuzdi (6.13-rasm). U har bir ko'rsatmani alohida varaqqa yozdi va- kerakli tartibda ustma-ust bir to'pga terdi. Shundan keyin algoritmi- endi **dastgacha** ko'chiraman deb turganida shumtaka Amon yugurib- kelib barcha varaqlarni aralashtirib yubordi. Qizig'i shundaki,- shundan keyin ham Robotni *D* harfli kvadratga olib keladigan- algoritm hosil bo'ldi. Buning sababini tushuntirib bering.

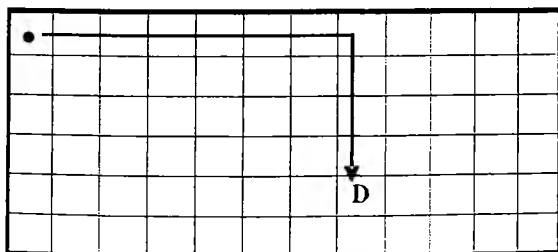


6.13-rasm.

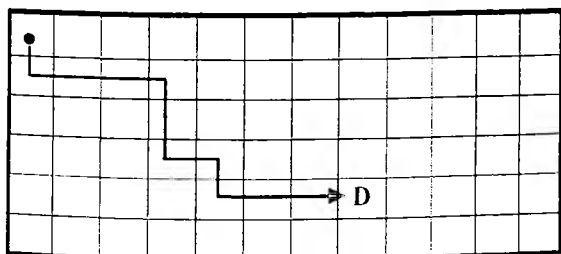
Yechim. Sunnatillaning algoritmidagi 7 ta **o'ngga** ko'rsatmasi va 4 ta **quyiga** ko'rsatmasi bo'lgan. 7 ta **o'ngga** ko'rsatmasi va 4 ta **quyiga** ko'rsatmasi bo'lgan har qanday yo'l baribir *D* harfli kvadratga olib keladi.

6.4-masha

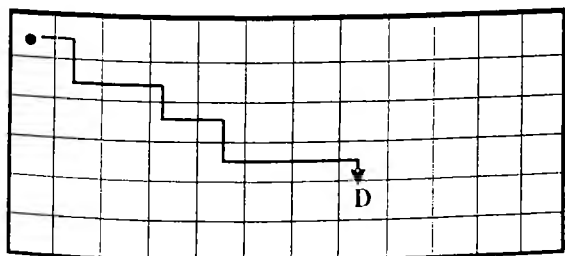
6.14-6.15-6.16-rasmlardagi Robotning chapga va quyiga necha qadam yurganini sanang.



6.14-rasm.
Sunnatillaning algoritmi.



6.15-rasm.
Kartochkalar aralashib ketdi.



6.16-rasm.
Kartochkalar o'aytadan aralashtirildi.

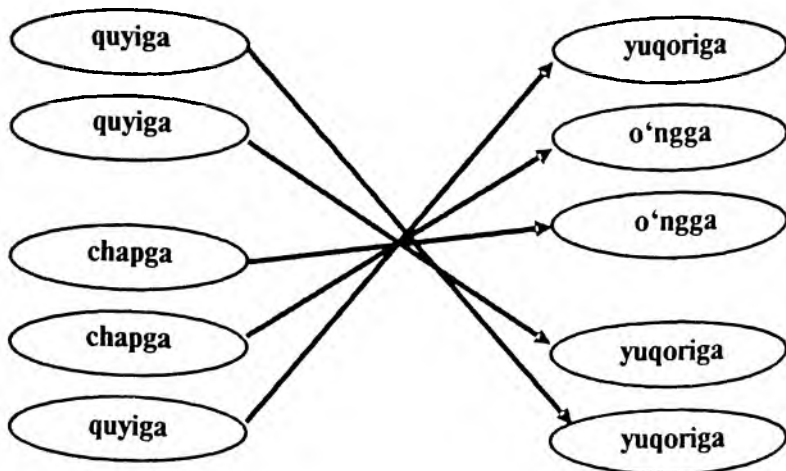
Robot chalkashib ketgan labirintda aylanib yurib «xazina» (bo'yalgan kvadrat) topib oldi. U quyidagicha yurgan edi:

quyiga
quyiga
o'ngga
quyiga
o'ngga
o'ngga
yuqoriga
chapga
yuqoriga
o'ngga
yuqoriga
chapga
chapga
quyiga

6.15-masala

Robot qanday qilib labirintdan chiqib ketadi, ya'ni oxirgi kvadratdan dastlabki kvadratga aniq teskari yurib bora oladi?

Yechim. Robotning yurgan yo'lini chizib, masalani yechish qulay. Lekin busiz ham yechish mumkin. Bunda bizga avvalgi boblarda ko'rilgan (eslab ko'ringchi) mulohazaning umumiy usuli yordam beradi. Robot qilgan oxirgi qadam **quyiga** edi. Demak, orqaga qaytayotganda birinchi qadam **yuqoriga** bo'ladi. Oxirigida bitta oldingi qadam **chapga** edi, demak, ikkinchi qadam **o'ngga** bo'ladi va hokazo. Natijada quyidagi algoritmlarni hosil qilamiz



6.5-masha

Algoritmi mustaqil ravishda to'liq yozib chiqing.

Bu usulning mohiyatini yana bir bor takrorlaymiz. Har bir ko'rsatmani aksi bilan almashtiramiz: **o'ngga** o'rniga **chapga** yoziladi, **chapga** o'rniga **o'ngga**, **yuqoriga** o'rniga **quyiga** va nihoyat **quyiga** o'rniga **yuqoriga** yoziladi. Endi aks ko'rsatmalarni teskari tartibda yozib chiqilsa bo'ldi.

Bu usulni oddiy hayotiy misolda ham ko'rish mumkin. Shunday ko'rsatmani qaraymiz:

paypog'ingni kiy

Aks ko'rsatma quyidagicha bo'ladi:

paypog'ingni yech

Xuddi shunday

tuflicingni kiy

ko'rsatmasiga

tuflicingni yech

ko'rsatmasi aks bo'ladi.

«Oyoq kiyimini kiyish» algoritmini yozamiz:

paypog'ingni kiy

tuflicingni kiy

Endi bu algoritimga teskari «Oyoq kiyimini yechish» algoritmi quyidagicha bo'ladi:

tuflicingni yech

paypog'ingni yech

Ko'rib turganingizdek, bu algoritmda ikkita aks ko'rsatma **teskari tartibda** yozilgan.

6.16-masala

Robotni bir kvadratdan ikkinchisiga olib boradigan (xohlasangiz, labirintda deb hisoblang) algoritmi yozing. Do'stingiz esa Robotni bosib o'tgan yo'lini oxiridan boshigacha aniq teskari yo'nalishda harakatlantiradigan algoritmi tuzsin. Siz ham do'stingiz tayyorlagan algoritmi uchun ham shu ishlarni bajaring.

6.17-masala

Robot maydonida devorlar yo'qligi ma'lum. Robot quyidagi algoritmi bajardi:

quyiga

quyiga

quyiga

o'ngga

o'ngga
 quyiga
 o'ngga
 yuqoriga
 yuqoriga
 chapga
 chapga

Bu harakatlardan keyin Robot A kvadratdan B kvadratga o'tad. Shunday algoritm tuzingki, Robot eng qisqa yo'l orqali (ya'ni eng kam qadamda) B kvadratdan A kvadratga qaytsin. Algoritmni rasm chizmasdan tuzishga harakat qiling.

Yechim. Bizning algoritmimizda 4 ta **quyiga** ko'rsatmasi va 2 ta **yuqoriga** ko'rsatmasi bor. Demak, natijada Robot 2 ta katak **quyiga** suriladi. Boshlang'ich holatga qaytish uchun 2 ta **yuqoriga** ko'rsatmasini bajarish yetarli. Xuddi shunday, algoritmimizda 3 ta **o'ngga** ko'rsatmasi va 2 ta **chapga** ko'rsatmasi bor. Boshlang'ich holatga qaytish uchun 1 ta **chapga** ko'rsatmasini bajarish yetarli. Demak, Robotni B katakdan A katakka siljitadigan eng sodda algoritmda 2 ta **yuqoriga** ko'rsatmasi va 1 ta **chapga** ko'rsatmasi bo'lish yetarli. Maydonda devorlar bo'lmagani uchun bu ko'rsatmalarni ixtiyoriy tartibda, ya'ni, masalan, quyidagicha yozish mumkin:

yuqoriga
 yuqoriga
 chapga

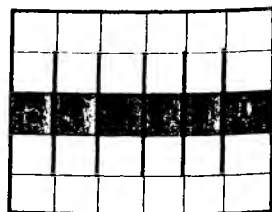
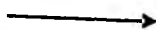
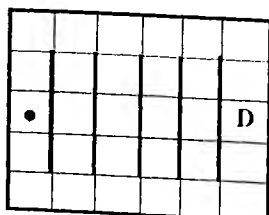
6.6-masha

6.17-masaladagi algoritmlarni rasmini ikki xil rangli qalamda chizing.

Takrorlanuvchi holatlar

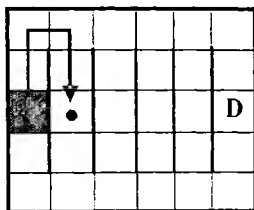
6.18-masala

Robot maydonida chapdagi rasmda ko'rsatilganidek, devorlar bor (6.17-rasm). Robotni D katakka o'tkazish va shu bilan birga ikkinchi rasmda ko'rsatilganidek, devorlar orasidagi kataklarni bo'yash kerak.



6.17-rasm.

Yechim. Rasmdan ko‘rinib turibdiki, Robot bir xil ko‘rsatmalarni bir necha bor takrorlashi kerak bo‘ladi: katakni bo‘yash, keyin devorni aylanib o‘tish (6.18-rasm):



6.18-rasm.

Bu ko‘rsatmalarni protsedura ko‘rinishida yozish qulay:

**PROT aylanish
BOSHLANISH**

bo‘ya
yuqoriga
yuqoriga
o‘ngga
quyiga
quyiga

TAMOM

Endi protsedurani 5 marta takrorlasak bo‘ldi, nima dedingiz? Lekin agar shunday yo‘l tutsak kamchilikka yo‘l qo‘yar ekanmiz. Algoritm 5 ta katakni bo‘yab *D* katakni bo‘yamas ekan. Bu kamchilikni osongina tuzatish mumkin:

TAKRORLANSIN 5 MARTA

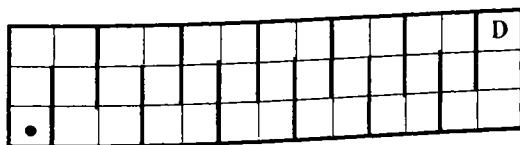
aylanish

TAMOM

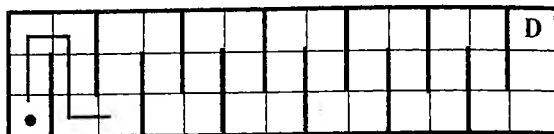
bo‘ya

6.19-masala

Robotni *D* katakka, ya‘ni chap quyi katakdan o‘ng yuqori katakka o‘tkazuvchi algoritm tuzing (6.19 rasm).



6.19-rasm.



6.20-rasm.

Yechim. Bu masalada ham bir xil ko‘rinishda takrorlanadig yo‘lning qismi bor. Bu qism 6.20-rasmda ko‘rsatilgan. Har doiml dek, yo‘lning bu qismini protsedura ko‘rinishida yozish qulay:

PROT so‘roq

BOSHLANISH

yuqoriga

yuqoriga

o‘ngga

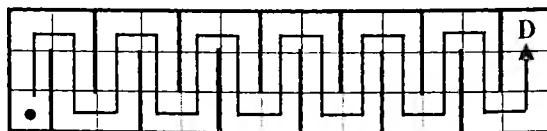
quyiga

quyiga

o‘ngga

TAMOM

Agar protsedura 6 marta takrorlansa, Robot yo‘lning oxirgi ikki yurishsiz 6.21-rasmga tasvirlanganidek bosib o‘tadi.



6.21-rasm.

Shuning uchun bizga kerakli algoritm quyidagicha ko‘rinish bo‘ladi:

TAKRORLANSIN 6 MARTA

so‘roq

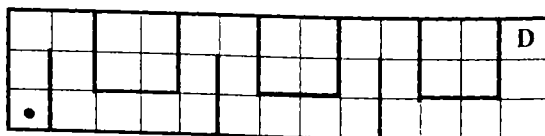
TAMOM

yuqoriga

yuqoriga

6.20-masala

Robotni D katakka olib keluvchi algoritm tuzing (6.22-rasm)



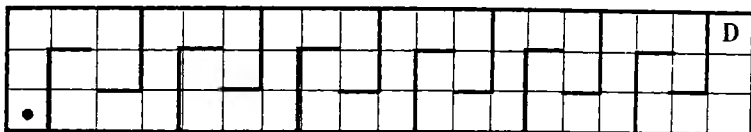
6.22-rasm.

Yo'llanma. Avvalgi masalalar kabi takrorlanuvchi qismlarni topib protsedura ko'rinishida tashkil qilishni maslahat beramiz.

Shu kabi mulohaza keyingi masalaga ham o'tadi, lekin undagi labirint ancha murakkablashgandir.

6.21-masala

Robotni *D* katakka olib keluvchi algoritm tuzing (6.23-rasm).



6.23-rasm.

Masalani mustaqil yecha olishingizga ishonamiz.

TAKRORLANSIN – MARTA tuzilmasini qo'llash qulay bo'lgan yana bir holat: Robot uzoq yo'lga ketmoqda. Bu holda **o'ngga** ko'rsatmasini ko'p marta yozgandan ko'ra takrorlash tuzilmasini qo'llash qulay.

6.22-masala

Robot maydoni 6x12 kvadratli to'g'ri to'rtburchakdan iborat. Robot chap yuqori burchakdagi katakda turibdi (6.24-rasm). Chegara yoqalab yurib barcha chegaraviy kataklarni bo'yash algoritmini tuzish talab etiladi.

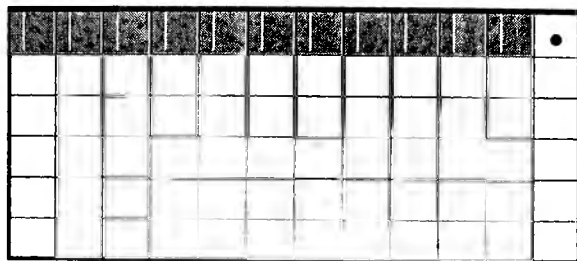


6.24-rasm.

Yechim. Avval yuqori chegara bo'ylab yuramiz (e'tibor qiling, o'zimizni Ijrochi Robotni o'rniga qo'yib ko'rayapmiz). Kataklar soni 12 ta, shuning uchun turgan katagimizni hisobga olmagan holda 11 ta qadam yuramiz:

TAKRORLANSIN 11 MARTA
bo'ya

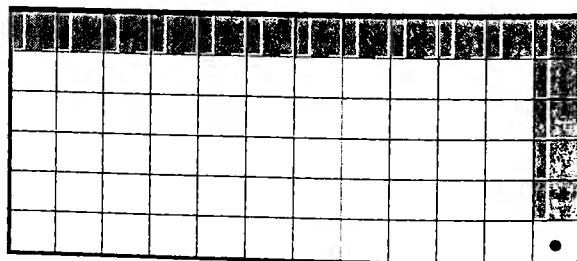
o'ngga
TAMOM
 O'ng yuqori katak bo'yalmasdan qoldi (6.25-rasm).



6.25-rasm.

Bu katak o'ng chegara bo'ylab yurganimizda bo'yaladi (6.26-rasm):

TAKRORLANSIN 11 MARTA
bo'ya
quyiga
TAMOM



6.26-rasm.

6.7-masha

Algoritmi mustaqil ravishda to'liq yozib chiqing.

6.23-masala

Robot maydoni 6x12 bo'lgan to'g'ri to'rtburchakdan iborat. Robot chap yuqori burchakdagi katakda turibdi (6.24-rasmga qarang). Barcha kataklarni bo'yash algoritmini tuzing.

Yechim. Maydonning barcha kataklarini juda ko'p usul bilan aylanib chiqish mumkinligini tushunish qiyin emas. Robot o'z turgan gorizontaldagi kvadratlarni bo'yashi uchun avval yo'lakni bo'ya protsedurasini yozamiz:

PROT yo'lakni bo'ya

BOSHLANISH

TAKRORLANSIN 11 MARTA

bo'ya

o'ngga

TAMOM

bo'ya

TAMOM

To'g'ri to'rtburchakning gorizontol yo'lagi 12 ta katakdan iboratligiga e'tibor bering, shuning uchun Robot o'ngga 11 marta surilishi kerak. Lekin bo'yalishi kerak bo'lgan kataklar 12 ta. Shuning uchun protseduraga siklga kirmagan yana bitta **bo'ya** ko'rsatmasini qo'shdik.

Endi soddagina bo'lgan **orqaga qayt** protsedurasini yozamiz:

PROT orqaga qayt

BOSHLANISH

TAKRORLANSIN 11 MARTA

chapgga

TAMOM

TAMOM

Yuqoridagilarga asosan **yo'lakni bo'ya** va **orqaga qayt** protsedurasini tuzamiz:

PROT yo'lakni bo'ya va orqaga qayt

BOSHLANISH

yo'lakni bo'ya

orqaga qayt

TAMOM

Endi to'liq algoritmni yoza olamiz va u quyidagicha:

yo'lakni bo'ya va orqaga qayt

TAKRORLANSIN 5 MARTA

yo'lakni bo'ya va orqaga qayt

TAMOM

6.8-masha

Quyidagi algoritmni vozish nima uchun xato bo'lardi?

TAKRORLANSIN 6 MARTA

yo'lakni bo'ya va orqaga qayt

TAMOM

Yo'llanma. Robot maydonida **yo'lakni bo'ya** va **orqaga qayt** protseduralari bajarilgach, nima bo'lishini ko'rish uchun rasm chizishni tavsiya etamiz.

«Robot maydonida nima bo‘lishini ko‘rish» savolida ikkita savolni ko‘rish kerak:

- Robot qayerda joylashgan bo‘ladi?
- Qaysi kataklar bo‘yaladi?

6.9-masha

Robot mavdon kataklarini boshqacha tartibda avlanib chiqadigan algoritm tuzing.

6.24-masala

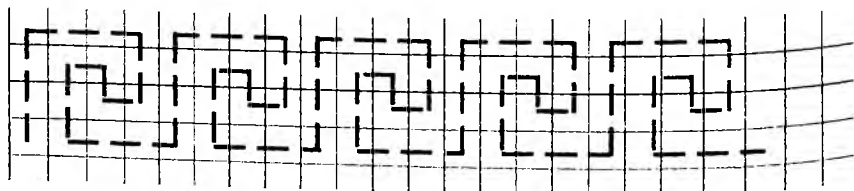
Robot quyidagi algoritmni bajarganda harakat qiladigan yo‘lini katak varaqqa chizing:

TAKRORLANSIN 5 MARTA

yuqoriga
yuqoriga
yuqoriga
o‘ngga
o‘ngga
o‘ngga
quyiga
quyiga
chapga
yuqoriga
chapga
quyiga
quyiga
quyiga
o‘ngga
o‘ngga
o‘ngga

TAMOM

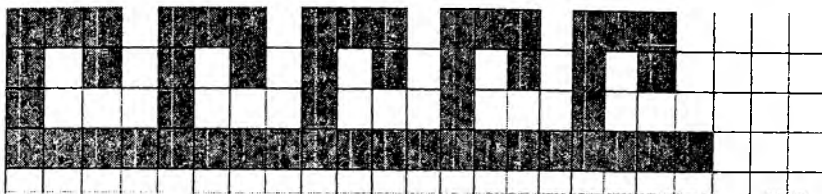
Javob. Robot yuradigan yo‘l 6.27-rasmda ko‘rsatilgan.



6.27-rasm.

Qadimgi yunonlar 6.27-rasmda tasvirlangan naqshni juda yaxshi ko'rishardi. Ular bu naqshdan binolarni, vazalarni, kiyimlarni va boshqa narsalarni bezatishda juda ko'p foydalanishgan. Yunonlar bu naqshni qirg'og'i ilonizi bo'lgan Meandr daryosiga qiyoslab **meandr** deb atashgan.

Meandrni Robot yordamida kataklarni boshqacha bo'yab ham tasvirlash mumkin (6.28-rasm).



6.28-rasm.

6.25-masala

6.28-rasmdagi meandrni tasvirlovchi algoritm tuzing.

Qiziqarli masalalar

Biz hozirgacha yechgan masalalarda maydondagi holat to'liq berilgan edi. Biz Robot qayerda turganini, u qayoqqa yurishi kerakligini, maydon o'lchami qandayligini, devorlar maydonning qayerida joylashganini bilar edik. Bir so'z bilan aytganda, nimani bilish mumkin bo'lsa, hammasini bilar edik. Shuning uchun ham algoritmni tuzish oson kechardi: har bir qadam oldindan ma'lum. Faqatgina bitta qiyinchilik bor edi, u ham bo'lsa algoritm ba'zan uzun bo'lib ketardi.

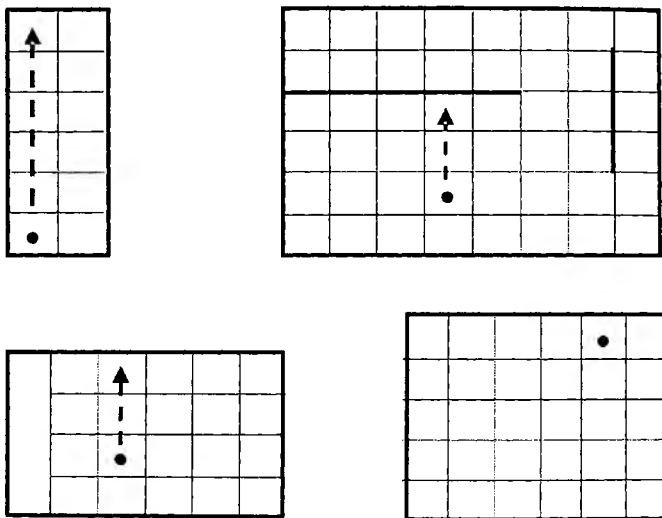
Algoritm tuzish biror narsa noma'lum bo'lganda qiziqarliroq kechadi. U holda Robotning o'zi vaziyatga moslashishga va sharoitdan kelib chiqib, o'zini turlicha tutishga majbur bo'ladi. Bu holda siz bilan bizga esa dasturchi sifatida har qanday holatlarni hisobga olishga to'g'ri keladi.

6.26-masala

Biz Robot qayerda turganini bilamiz. U yuqorida joylashgan eng yaqin devorning oldiga borib to'xtashi kerak.

1-sharh

6.29-rasmda har xil bir nechta maydon tasvirlangan bo'lib, Robot borishi kerak bo'lgan katak strelkaning uchi yordamida ko'rsatilgan.



6.29-rasm.

Oxirgi to'rtinchi holatda yurishga joy yo'q – Robot yuqori devor oldida turibdi. Masalaning murakkabli shundaki, Robot yuqoriga nechta qadam yura olishini avvaldan ayta olmaymiz: 2 tami, 5 tami, 0 tami yoki bo'lmasa, 100 000 tami (agar maydon yetarlicha katta bo'lsa). Boshqacha aytganda, Robot «toki imkoni bor ekan yuqoriga» yuradi.

2-sharh

Har qanday masalani ikki dasturchi uchun o'yin sifatida qarash mumkin. Avval birinchi dasturchi Robot uchun algoritm tuzadi – Qo'yilgan masalaning yechimi. Keyin ikkinchi dasturchi Robot maydonini quradi – Masala shartiga mos. U maydon o'lchamlarini tanlaydi, devorlar chizadi va xohlagan joyiga Robotni joylashtiradi. Shundan keyin Robot birinchi dasturchi tuzgan algoritmni bajaradi va u to'g'ri natija berishi shart!

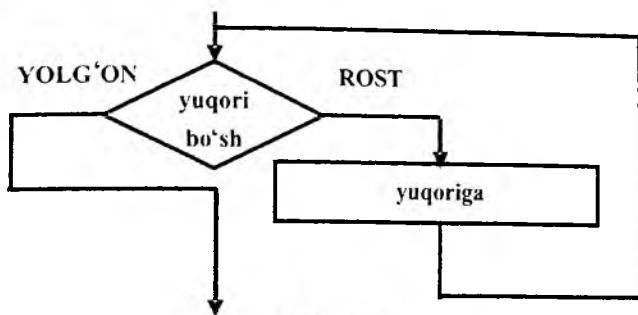
Yechim. Masalamizning yechimi hayron qolarli darajada sodda. Yechimga yo'llanma berilgan "toki imkoni bor ekan yuqoriga" sharhining ichida yotibdi.

TOKI yuqori bo'sh BAJAR

yuqoriga

TAMOM

Ba'zan algoritmni blok-sxema ko'rinishida tasvirlash zarar qilmaydi, aksincha, foydali bo'ladi, yuqoridagi algoritmning blok-sxemasi quyidagicha:



6.30-rasm.

6.27-masala

Robot devor bilan o'ralgan to'g'ri to'rtburchak ichida joylashgan, maydon ichida boshqa devorlar yo'q. Robotning qaysi katakda joylashgani noma'lum. Robotni o'ng yuqori burchakka olib keluvchi algoritm tuzing.

Yechim. Oldingi masalaning yechimini bilganimiz uchun bu masala oson yechiladi:

TOKI yuqori bo'sh BAJAR

yuqoriga

TAMOM

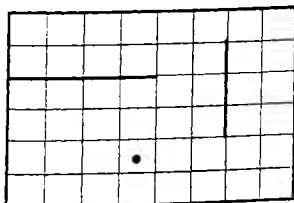
TOKI o'ng bo'sh BAJAR

o'ngga

TAMOM

6.28-masala

Faraz qilaylik, Robot maydonida ichki devorlar bor ekanligini yashirishgan bo'lsin (6.31-rasmdagi kabi). Robotni o'ng yuqori burchakka olib keluvchi algoritm tuzing.



6.31-rasm.

Devorlarni bunday joylashishida yuqoridagi algoritm masalani hal eta olmaydi. Lekin bu algoritmning o'ziga yarasha muhim bir afzallik tomoni bor. Agar avvalgi masalada biz maydon o'lchamini va Robot turgan joyi oldindan bilganimizda algoritmni «soddaroq» qilib yozardik:

TAKRORLANSIN 4 MARTA

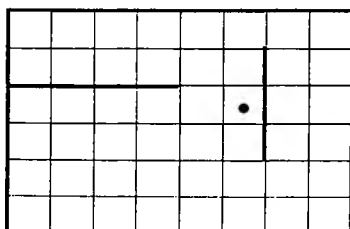
yuqoriga

o'ngga

TAMOM

o'ngga

Bu algoritmni hozirgi masalaga qo'llab bo'lmaydi, chunki Robot ko'zda tutilmagan devorga urilib sochilib ketadi. Avvalgi masala yechimidagi algoritm esa halokatga olib kelmaydi. Chunki har bir qadam qo'yishdan avval Robot devor bor-yo'qligini tekshirib ko'radi va 6.32-rasmdagi holatga keladi.



6.32-rasm.

Yechim. Devorlar holatini bilganimiz uchun avvalgi masala yechimini ikki marta qo'llaymiz, ya'ni algoritm quyidagicha bo'ladi:

TAKRORLANSIN 2 MARTA

TOKI yuqori bo'sh BAJAR

yuqoriga

TAMOM

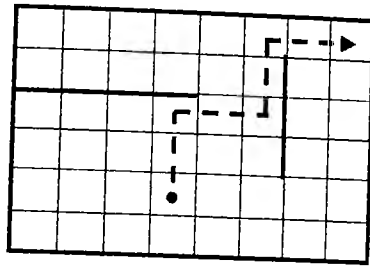
TOKI o'ng bo'sh BAJAR

o'ngga

TAMOM

TAMOM

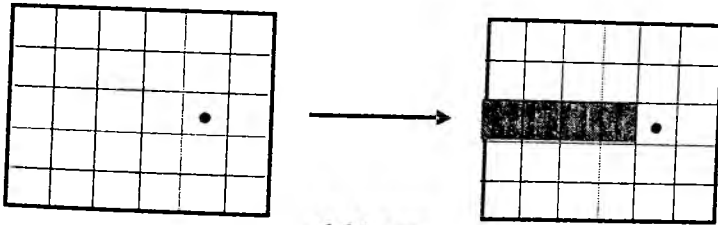
Bu algoritm natijasida Robot 6.33-rasmdagi yo'l bilan yuqori o'ng burchakka boradi. Birinchi takrorlanishda Robot avval **TOKI yuqori bo'sh BAJAR** protsedurasiga ko'ra yuqoriga ikki qadam yuradi va devor oldida to'xtaydi. Keyin **TOKI o'ng bo'sh BAJAR** protsedurasiga ko'ra o'ngga ikki qadam yuradi va devor oldida to'xtaydi (6.32-rasm). Endi ikkinchi takrorlanish boshlanadi va Robot ikkala protsedurani navbati bilan bajarib, yuqori o'ng burchakka boradi. To'g'ri to'rtburchakning yuqori va o'ng devorlari Robotni siljitgani qo'ymaydi. Agar takrorlanishlar soni 2 dan ziyod berilsa ham Robot sochilib ketmasdan, yuqori o'ng burchakka borib to'xtagan bo'lardi!



6.33-rasm.

6.29-masala

Robot va chap devor orasidagi barcha kataklarni bo‘yang va Robotni boshlang‘ich holatiga qaytaring (6.34-rasm).



6.34-rasm.

Yechim. Avval ko‘rilgan masalalardan devorgacha borishni bilamiz, faqatgina bo‘yashni qo‘shamiz, xolos:

TOKI chap bo‘sh BAJAR

chapga

bo‘ya

TAMOM

Endi qanday qilib avval Robot turgan katakka qaytishni o‘ylaymiz. Yodingizda bo‘lsa, Robot katak bo‘yalgan yoki bo‘yalmaganligini tekshirishni bilar edi, shunga ko‘ra yozamiz:

TOKI bo‘yalgan BAJAR

o‘ngga

TAMOM

Masala to‘liq hal bo‘ldi.

E‘tiboringizni shunga qaratamizki, bo‘yalgan kataklargina Robotning boshlang‘ich katagiga qaytarishga imkon berdi. Bo‘yalish shartisiz Robot qayerda to‘xtashni bilmagan bo‘lar edi. Biz-ku, Robot avval qayerda turganligini esladik, lekin Robot eslay olmaydi — uning xotirasi yo‘q. Eslatib o‘tamiz, Robot — faqatgina Ijrochi: u bob boshida keltirilgan o‘zi biladigan 5 ta ko‘rsatmani bajara oladi va 5 ta shartni tekshira oladi, bechorani boshqa hech narsasi yo‘q!

Shunday qilib, biz foydali usulni kashf etdik. **Robot** o'ziga xos **xotirasi** sifatida **bo'yalgan kataklarni** ishlatishi mumkin ekan. Bo'yalgan kataklar, masalan, Robot qachonlardir shu katakda bo'lganligini bildiradi. Siz ham g'orda adashib qolsangiz, shu kabi ish tutishingiz mumkin. Siz bir joyni ikkinchisidan farqlash uchun g'orda har xil belgilar qoldirishingiz mumkin. To'g'ri, Robotga nisbatan sizda imtiyoz ko'proq — Robot bitta bo'yashni bilsa, siz har xil belgilashlardan foydalana olasiz!

6.30-masala

Robotning maydoni noma'lum o'lchovli to'g'ri to'rtburchak shaklida. Robot chap yuqori katakda turibdi. Maydonning chegarasi yoqalab yurib chegaraviy kataklarni bo'yab chiqish talab etiladi.

Yechim. Sezgan bo'lsangiz, bu masala 6.22-masalani deyarli so'zma-so'z takrorlaydi. U masalada yuqori chegara uzunligi 12 ta katak bo'lgani bois, yuqori chegara bo'yab 11 qadam yurgan edik. Endi esa chegaralar uzunligini bilmaymiz va o'ngga nechta qadam yurish kerak bo'lsa, shuncha yurmoqchimiz. Buning uchun 4 xil yo'nalishdagi bo'sh kataklar uchun **TOKI — BAJAR** tuzilmasidan foydalanish kerakligini bilamiz. Masalan:

Avvalgi dasturda
TAKRORLANSIN 11 MARTA
bo'ya
o'ngga
TAMOM

Yangi dasturda
TOKI o'ng bo'sh BAJAR
bo'ya
o'ngga
TAMOM

6.10-masha

Algoritmni mustaqil oxiriga vetkazing. Tuzgan algoritmingiz 6.35-rasmda berilgan 1x1 o'lchamli to'g'ri to'rtburchak uchun qanday ishlaydi?

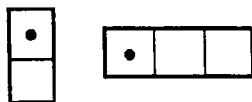


6.35-rasm.

Yechim. Ha, bu variantni biz ko'zda tutmaganmiz shekilli. Bunday maydonda Robotga bir qadam ham qo'yishi shart emas. Hamma kamchilik shundaki, u birorta ham katakni bo'yamaydi, chunki **TOKI — BAJAR** orasida yotgan shart hech qachon **ROST** qiymat qabul qilmaydi. Lekin, shu birgina katak ham chegaraviy hisoblanadi va demak, masala shartiga ko'ra bo'yalishi kerak. O'z algoritmingizni tahlil qilib ko'ring. Agar shunday tushunmovchilik sizda ham uchrasa, u holda algoritm oxiriga bitta **bo'ya** ko'rsatmasini qo'shib qo'ying.

6.11-masha

Tuzgan algoritmingiz 6.36-rasmda berilgan mavdonlarda qanday ishlashini tekshiring.



6.36-rasm.

6.31-masala

Robot maydoni – ichki devorlari bo'lmagan to'g'ri to'rtburchak. Robot chap yuqori katakda turibdi. Maydonning barcha kataklarini bo'yash talab etiladi.

Yo'llanma. 6.23-masalaning yechimini shu masala yechimiga aylantirish mumkin.

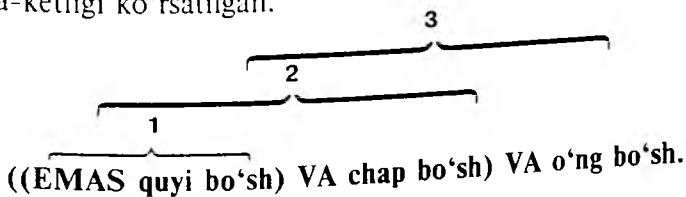
Birikkan shartlar

Endi sizning Robot bilan ishlash tajribangiz bor. Balki sezgandirsiz, biz ba'zi sodda narsalarni tekshirishni bilmaymiz. Biz **yuqori bo'sh** shartining rostligini tekshira olamiz. Yoki Robot yuqori devor oldida turgani, ya'ni yuqori bo'sh emasligi rostligini tekshira olamiz. Lekin biz Robot yuqori chap katakda turganini, ya'ni **yuqori bo'sh emas** va **chap bo'sh emas** shartini rostligini tekshirishimiz qiyin. Ammo Robot ajoyib-g'aroyib labirintlar ichida yurganda shartlarning har xil birikmalarini tekshirish kerak bo'lishi mumkin.

Avvalgi bobda ko'rdikki, mantiqiy amallar juda murakkab shartlarni ham biriktirish imkonini beradi, masalan, «Robot quyi chegarada turibdi, lekin burchakda emas» kabilarni.

Masalan, Robot chap yuqori katakda turish sharti quyidagicha yoziladi: **yuqori bo'sh emas VA chap bo'sh emas**.

Eslatib o'tamiz, murakkab mantiqiy birikmalarda qavs qo'llanilib, arifmetik amallardagi kabi avval ichki qavslar ichidagi Shart qiymati hisoblanadi. Masalan, quyida amallar bajarilish ketma-ketligi ko'rsatilgan:



6.12-masha

6.37-rasmdagi har bir rasm uchun quyidagi shartlarning har birini rost voki volg'onligini tekshiring.



6.37-rasm.

yuqori bo'sh VA chap bo'sh
yuqori bo'sh YOKI o'ng bo'sh
EMAS yuqori bo'sh

6.32-masala

Robot maydoni – ichki devorlari bo'lmagan to'g'ri to'rt-burchak. Robotni devorlardan uzoqlashtiradigan (agar mumkin bo'lsa, ya'ni maydonning bo'yi va eni 3 ta katakdan kam bo'lmasa) algoritmi tuzing (agar maydon bo'yi yoki eni 3 ta katakdan kam bo'lsa, Robot xohlaganicha harakat qilishi mumkin, faqat sochilib ketmasa bo'ldi).

Yechim. Bu yerda yechimning chorak qismini keltiramiz. Robotni chap devordan uzoqlashishga o'rgatamiz:

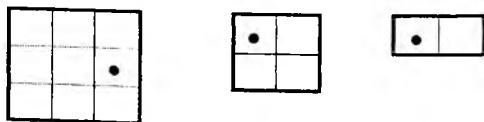
AGAR (EMAS chap bo'sh) VA o'ng bo'sh
o'ngga

TAMOM

Ko'rib turganingizdek, Robot chap devor yonida turibdimi-yo'qmi va o'ngga bir qadam yurish mumkin yoki yo'qligini tekshirdik.

6.13-masha

Algoritmi mustaqil vakunlang. Tuzgan algoritmingiz 6.38-rasmda berilgan rasmlardagi holatlarda qanday ishlashini tekshiring.

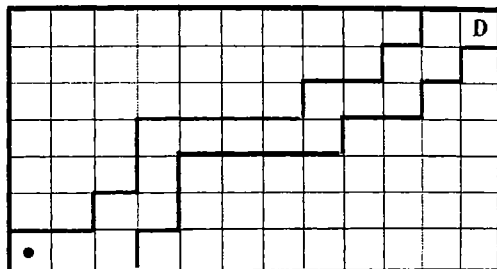


6.38-rasm.

Birikkan shartli algoritmlar

1. Yo'lak bo'ylab yurish. Robot yo'lak bo'ylab yurib maydonning bir burchagidan boshqa burchagiga o'tishi kerak. Yo'lakning aniq shakli ma'lum emas. Faqat uning kengligi 1 ta katakligi

va chap-quyidan o'ng-yuqori yo'nalishda cho'zilib ketgani ma'lum. Bunday yo'lakning namuna ko'rinishi 6.39-rasmda tasvirlangan.



6.39-rasm.

Birinchi navbatda Robotning to'xtashi uchun nima xizmat qilishini aniqlab olaylik. Sayohat yuqori o'ng burchakda tugashi kerak. «Robot yuqori o'ng burchakda» sharti quyidagicha yoziladi:

(EMAS yuqori bo'sh) VA (EMAS o'ng bo'sh).

Avvalgi bobdan ma'lumki, bu shartga teskari "Robot hali burchakda emas" sharti mana bunday:

yuqori bo'sh YOKI o'ng bo'sh.

Robot hali burchakda bo'lmasa, harakatda bo'lishi shart. Demak, bizning algoritmimiz bunday ko'rinishda boladi:

TOKI yuqori bo'sh YOKI o'ng bo'sh BAJAR

bir qadam yur

TAMOM

Endi qaysi yo'nalishda qadam tashlash kerakligini hal etamiz – javob aniq. Haqiqatan **TOKI – BAJAR** shartining o'zida yuqori yoki o'ng tomon albatta bo'sh ekanligi aytilgan. «**bir qadam yur**» protsedurasini quyidagicha yozish mumkin:

PROT bir qadam yur

BOSHLANISH

AGAR yuqori bo'sh

U HOLDA

yuqoriga

AKS HOLDA

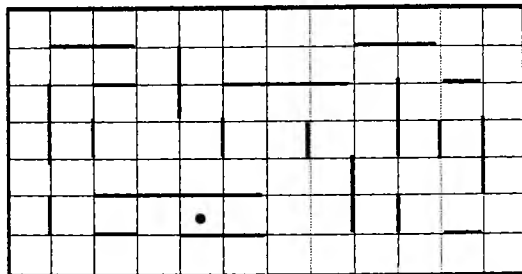
o'ngga

TAMOM

TAMOM

Mana, masalani hal qilib qo'ydik!

2. Labirintdan chiqish. Robot labirintning qayeridadir turibdi. Labirintning ichidagi devorlar kesmalardan iborat va ular bir-biri bilan va tashqi devorlar bilan kesishmaydi (6.40-rasm). Shunday algoritm tuzingki, uning bajarilishi natijasida Robot ixtiyoriy shu kabi labirintda yuqori o'ng burchakka borsin.



6.40-rasm

Oldingi masalada tuzilgan algoritm bu masalaning ham yechimini beradi.

6.14-masha

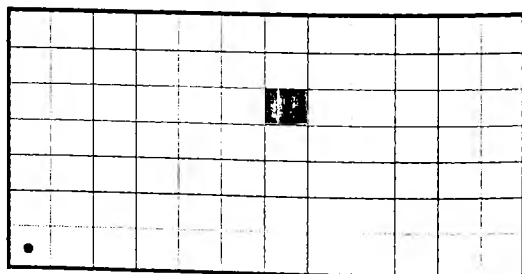
6.40-rasmdagi Robotning yurish yo'lini kuzatib boring.

6.15-masha

Robotni ko'rsatilgan joyga olib keladigan algoritm tuzing:

- a) yuqori chap burchakka;
- b) yuqori o'ng burchakka;
- d) quyi o'ng burchakka.

3. Xazina qidirish. Robot maydoni — ichki devorlarsiz to'g'ri to'rtburchak. Robot quyi chap burchakda turibdi. Maydonning qayeridadir xazina bor (bo'yalgan katak) — (6.41-rasm). Xazinani toping.



6.41-rasm.

Robot bo'yalgan katakka borganida to'xtashi, ungacha yurishi shart. Mulohaza qilib algoritmni quyidagicha boshlaymiz:

TOKI EMAS bo'yalgan BAJAR

...

TAMOM

Endi qanday harakat qilishi kerakligi haqida fikrlab ko'ramiz. Xazinani faqat bir yo'nalishda – pastdan yuqoriga qarab qidirish oson tuyulmoqda. Aniqrog'i, algoritm tuzuvchi biz uchun oson, Robotga esa ish ko'payadi! Yuqoriga oddiygina qadam tashlab bo'lmaydi, Robot devorga urilib sochilib ketadi. Bunday ish tutamiz:

AGAR yuqori bo'sh

U HOLDA

yuqoriga

AKS HOLDA

...

TAMOM

AKS HOLDA qismi nimani anglatadi? U Robot yuqori devorga borib taqalganini bildiradi. Bundan kelib chiqadiki, bu vertikalda xazinani topolmadik.

Endi orqaga qaytib, keyingi vertikalga o'tishimiz kerak. Uning algoritmi quyidagicha:

TOKI quyi bo'sh BAJAR

quyiga

TAMOM

AGAR o'ng bo'sh

U HOLDA

o'ngga

TAMOM

Barcha qismlarni yig'sak, kerakli algoritm tayyor bo'ladi.

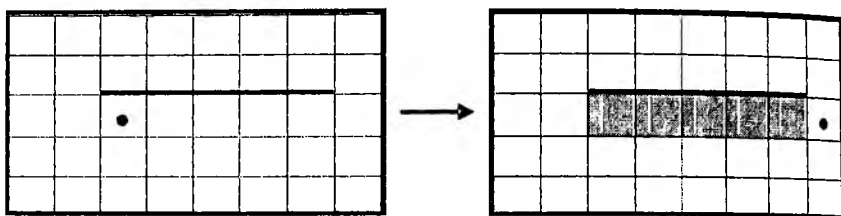
6.16-masha

Agar mavdonda xazina bo'lmasa, Robot nima qilarkin?

Mustaqil yechishingiz uchun ikkita masala keltiramiz.

6.33-masala

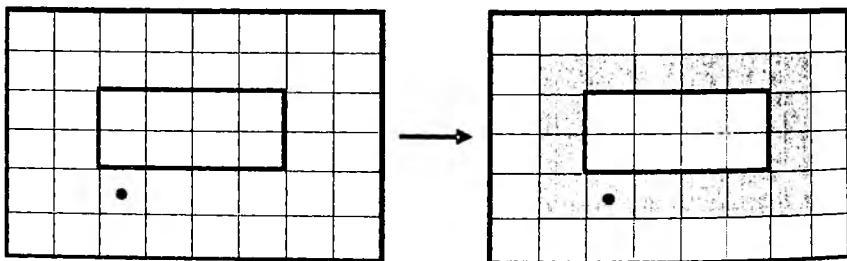
Robot gorizontaal devorning ostida bo'lib, uning chap chekkasida joylashgan. Devorning uzunligi noma'lum. Devor tagidagi yopishgan barcha kvadratlar bo'yalsin (6.42-rasm).



6.42-rasm.

6.34-masala

Robot maydonidagi ichki devorlar noma'lum o'lchamdagi to'g'ri to'rtburchakni tashkil etadi. Robot to'g'ri to'rtburchakning chap quyi burchagidan pastda turibdi. To'g'ri to'rtburchak atrofidagi unga yopishgan barcha kataklar bo'yalsin (6.43-rasm).



6.43-rasm.

Murakkab algoritm namunasi

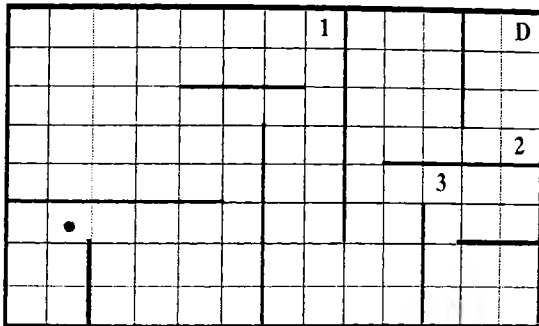
Sizga murakkab algoritmlardan hech bo'lmaganda bittasini ko'rish qiziqarli bo'lsa kerak. Bizning algoritmimizni malakali dasturchilar tuzgan algoritmlar bilan taqqoslab bo'lmaydi, albatta. Ularning algoritmi minglab satrlarni tashkil etadi, dasturchilar guruhi esa millionlab satr algoritm tuzishadi. Bizning algoritmimizni esa bitta sahifaga joylash mumkin. Shunga qaramay bizning algoritmimiz yetarlicha murakkabdир. Murakkabligi esa shu ma'nodaki, taklif qilayotgan masalamizni yechish uchun tajribali dasturchiga ham anchagina o'ylash, fikrlash zarur bo'ladi.

6.35-masala

Robot 6.44-rasmda tasvirlangani kabi labirintning qayeridadir turibdi. Labirintning ichki devorlari kesmalardan iborat bo'lib. o'zaro bir-biriga tegmaydi, lekin tashqi devorga tegib turishi mumkin. Robotni yuqori o'ng katakka o'tkazuvchi algoritm tuzilsin.

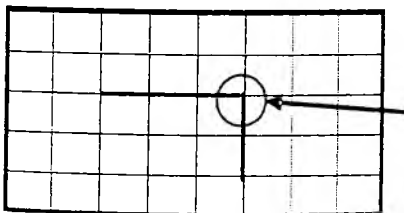
3-sharh

Avvalgi bo'limdagi Robotni labirintda harakatlantirish algoritmidan foydalanish kerakli natijaga olib kelmaydi. Rasmdagi holda bu algoritmni bajargan Robot 1 raqami yozilgan katakka kelib xuddi kerakli burchakka borgandek to'xtab qoladi.



6.44-rasm.

Quyidagicha ichki burchaklar yo'q:



Bunday ichki burchaklar
bo'lishi taqiqlangan

Avvalgi bo'limdagi algoritmni davom ettirishga harakat qilish, ya'ni tahrir qilib Robotni vertikal devorlarni aylanib o'tishga o'rgatish mumkin. Bu yerda boshqa muammo kutib turibdi. Robot qachon to'xtashi kerakligi noaniq. Haqiqatan Robot D katakka kelganidan keyin «navbatdagi devorni aylanib o'tish» uchun 2 tartib raqamli katak tomon yura boshlaydi. Lekin inson tilida aytish mumkin bo'lgan «navbatdagi devorni aylanib o'tishni ilojisi bo'lmadi» degan iborani Robot tilida yozish ancha murakkab.

Yechim. Masalani yechishni boshidan boshlaymiz. Ish rejamiz quyidagicha:

- yuqori chegaraga qarab yuramiz;
- unga yetib borgach o'ng chegara tomon yuramiz.

Lekin Robotga shart qo'yamiz: faqat yuqoriga qarab yurilsin, quyiga qarab hech qanday qadam qo'yilmasin!

Yuqoriga o'tish joyini qidirishdan boshlaymiz. Qiyinchilik shundan iboratki, o'tish joyi qaysi tomondaligi noma'lum. Masalan, bizdagi rasmda o'tish joyi Robotdan o'ngda joylashgan. Agar Robot 3 raqami yozilgan katakda bo'lganda, o'tish joyi chaproqda bo'lgan bo'lardi.

Shunday yo'l tutamiz: avval Robotni chap devorga qarab yurgazamiz, oxiriga borgach, yuqoriga o'tish joyini topish uchun o'ngga qarab yuramiz.

Ya'ni:

PROT yuqoriga o'tish joyini top

BOSHLANISH

TOKI chap bo'sh BAJAR

chapgga

TAMOM

... (o'ngga yurish va yuqoriga o'tish joyini topish)

TAMOM.

Protsedurani to'ldirish (qavs ichidagi so'zlarni Robot ko'rsatmasiga aylantirish) uchun yozamiz:

TOKI o'ng bo'sh VA (EMAS yuqori bo'sh) BAJAR

o'nggga

TAMOM

Barcha protsedura davomida Robot faqat gorizontaal yo'nalishda yoki, o'ngga yoki chapgga siljiyotganiga e'tibor bering. Protsedura ikki holatdan birida to'xtaydi:

- yoki Robot o'ngdagi devorga yetib keldi, lekin o'tish joyini topolmadi: bu holda Robot yuqori chegaraga yetib kelgan bo'ladi;
- yoki yuqoriga o'tish joyi topildi; u holda yuqoriga bir qadam qo'yib protsedurani boshidan bajaradi, ya'ni yana yuqoriga o'tish joyini qidiradi.

Quyidagi algoritmgga kelamiz:

PROT yuqori chegaraga qarab yur

BOSHLANISH

yuqoriga o'tish joyini top

TOKI yuqori bo'sh BAJAR

yuqoriga

yuqoriga o'tish joyini top

TAMOM

TAMOM

Endi Robot yuqori chegarada turibdi. Nima qilish kerak? Shu usulda ish tutilsin! Shu usulda o'ngga yuramiz: avval o'ngga o'tish

joyini qidiramiz, topgach, o'ngga bir qadam qo'yamiz va yana o'tish joyini qidiramiz va hokazo.

Buning uchun yozilgan algoritmlarda yuqoriga o'rniga o'ngga yoziladi va yana ba'zi o'zgartirishlar qilinadi.

6.17-masha

Algoritmi oxirigacha mustaqil yozing. Unga mos blok-sxema tuzing.

6.18-masha

O'naga o'tish loyini quyidagi usullardan birida amalga oshirish mumkin:

a) avval quyida qarab borib, keyin quyida Yurish cho'ida o'tish loyi ajdiriladi;

b) avval quyida qarab borib, keyin quyida Yurish cho'ida o'tish loyi ajdiriladi.

Nima deb o'ylaysiz, bu ikkala usul ham bo'laveradimi yoki faqat bittasi to'g'ri keladimi?

Yo'llanma. Usullardan biri xato: u bizni 2 raqami yozilgan katakka olib keladi. E'tiborli bo'ling!

Nazorat savollari va toshiriqlar

1. *Ibrochi Robot andav ko'rsatmalarni tushunadi?*
2. *Ibrochi Robot andav shartlarni tekshira oladi?*
3. *Ibrochi Robot mavdoni haqida so'zlab bering.*
4. *Ibrochi Robot uchun INKOR holat yuz berishiqa misol keltiring.*
5. *Meandr nima?*
6. *Robot uchun birikkan shartlar haqida so'zlab bering.*
7. *Robot uchun «ot yurish» protsedurasini izohlab bering.*
8. *Robot uchun bo'yalgan kataklar mohiyatini misollar yordamida ochib bering.*
9. *Bobdagi barcha masholarni bajarig.*

Qo'shimcha masalalar

R-6.1. Robot quyidagi algoritmi balardi:

- | | |
|---|--------|
| 1 | o'ngga |
| 2 | quyida |
| 3 | quyida |
| 4 | o'ngga |
| 5 | quyida |
| 6 | cho'ga |

Katak varaada Robot uchun ixtivoriv boshlang'ich holatni tanlab, uni har bir ko'rsatmani bajarandan kevinigi ioviga ko'rsatmaning tartib raqamini vozib boring.

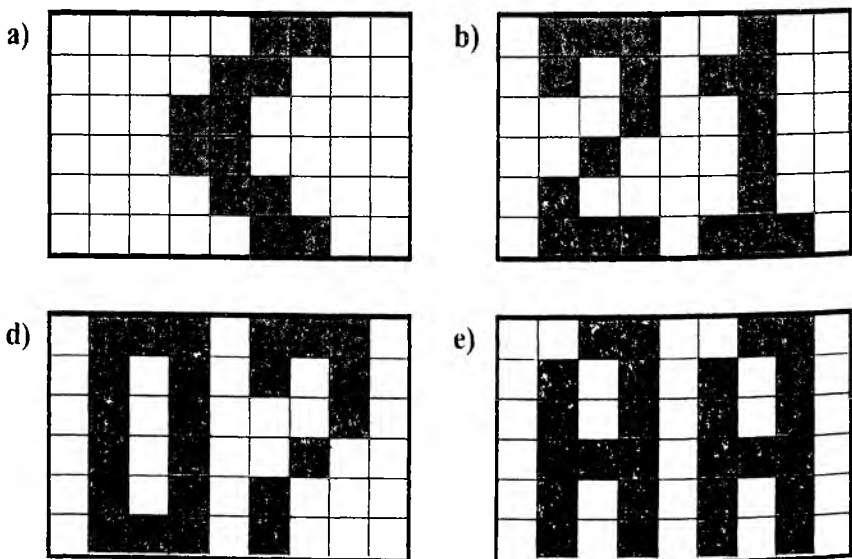
R-6.2. Robot bir algoritmnining har bir ko'rsatmasini bajarandan kevinigi ioviga ko'rsatmani tartib raqami vozilganda quvidagi 6.45-rasm hosil bo'ldi:

		8	7	6	5									
		9	2	3	4									
		10	1	14	15	16	17							
		11	12	13			18							

6.45-rasm.

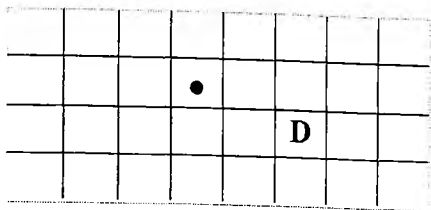
Robot bajaragan birinchi ko'rsatma **changa** edi. Robot bajaragan algoritmni vozib chiding.

R-6.3. Robot uchun shunday algoritm vozingki, ularni bajarilish natijasida quvidagi rasmlar hosil bo'lsin (6.46-rasm):



6.46-rasm.

R-6.4. Robotni uchta oadamda *D* katakka o'tkazuvchi algoritm vozing (6.47-rasm):



6.47-rasm.

- a) shu ishni bajaruvchi uch xil turli algoritm tuzing;
 - b) Robotni to'rtta qadamda *D* katakka o'tkazish mumkinmi?
 - d) Robotni beshta qadamda *D* katakka o'tkazish mumkinmi?
 - e) Robotni oltita qadamda *D* katakka o'tkazish mumkinmi?
- Javoblaringizni asoslab bering.

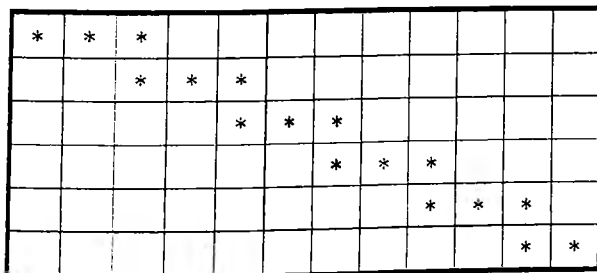
R-6.5. Robot quyidagi algoritmni bajardi:

o'uyiga
bo'va
chadga
o'uyiga
o'ngga
yuqoriga
bo'va
o'ngga
bo'va

- a) katak varada Robot uchun ixtiyoriv boshlang'ich holatni tanlab, uni har bir ko'rsatmasini bajarib chiqing;
- b) Robot bo'vagan kataklarni bo'vavdigan oisqa algoritm tuzing.

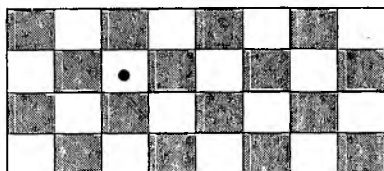
R-6.6. 8x8 katakli mavdonda harakat oila'otgan Robot mavdonni shaxmat doskasi kabi bo'vavdigan algoritm tuzing.

R-6.7. 6.48-rasmdagi zinapovani bo'vavdigan algoritm tuzing.



6.48-rasm.

R-6.8. Robot shaxmat kabi bo'valgan mavdonning oq katagida turibdi (6.49-rasm).



6.49-rasm.

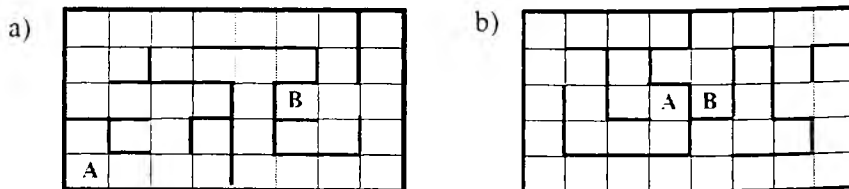
Robot oqidaqicha qadamlardan kevin oqvi rangdagi katakda turadi:

- a) 1 qadam;
- b) 2 qadam;
- d) 3 qadam;
- e) 4 qadam;
- f) 18 qadam;
- g) 124 qadam;
- h) 43 qadam

Natija qadamlar soni iuft ekanligiga qandav bog'langan? Javobingizni izohlang.

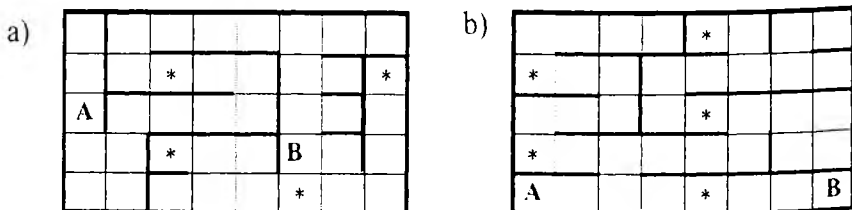
R-6.9. Robot $N \times M$ kvadratli to'g'ri to'rtburchak shaklidaqi devor bilan cheqaralangan mavdonda vuribdi. Mavdon ichida devorlar vo'a. Robot mavdonning har bir katagida faqat bir martadan bo'lib barcha kataklarni avlanib chiqib boshlang'ich katagiga qavta oladimi? Bu N va M sonlariga qandav bog'langan?

R-6.10. Robotni 6.50-rasmdagi labirint ichida A katakdan B katakka o'tkazuvchi algoritm tuzing.



6.50-rasm.

R-6.11. Robotni 6.51-rasmdagi labirint ichida A katakdan B katakka yulduzchali kataklarni bo'vab o'tkazuvchi algoritm tuzing.

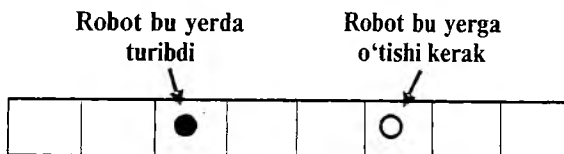


6.51-rasm.

O'zini chaqiradigan protseduralar

7.1-masala

Robot kengligi 1 ga teng va devorlar bilan o'ralgan gorizontal to'g'ri to'rtburchakning chap devoridan biror masofada turibdi. Robotni o'ng devordan shuncha masofada bo'lgan katakka o'tkazing (7.1-rasm).



7.1-rasm

Masala juda sodda bo'lib ko'rinadi. Haqiqatan, agar Robot bilan chap devor orasidagi, aytaylik, ikkita katakni, bilsak uni kerakli katakka jo'natish qiyin emas: Robot avval o'ng devorgacha boradi keyin ikkita chapga yuradi. Murakkabligi shundan iboratki, chap devorgacha bo'lgan masofa ixtiyoriy bo'lishi mumkin, bizda esa bu masofani o'lchash uchun biror usul ham, o'lchay olsak uning qiymatidan foydalana olish imkoniyati ham yo'q. Bu vaziyatdan chiqish, bunaqasi tez-tez bo'lib turadi, kutilmaganda sodir bo'ldi.

Yechim. Qo'yilgan masalani yecha oladigan protsedurani yozishga harakat qilamiz. Bu protsedurani **simmetriya** deb ataymiz. Avvalo bizga masalani hech bo'lmaganda Robot chap devor oldida turgan birgina holatda yechish mumkinligi yordam beradi. Bu holda Robot o'ng devor oldiga borib to'xtashi kerak:

PROT simmetriya

BOSHLANISH

AGAR EMAS chap bo'sh

U HOLDA

TOKI o'ng bo'sh BAJAR

o'ngga

TAMOM

AKS HOLDA

...

TAMOM

TAMOM

Endi Robotni chap yonida devor bo'lmasa, nima qilishi kerakligi haqida o'ylaymiz. Keling, chapga bir qadam tashlaymiz, Robot bilan chap devor orasidagi masofa qisqaradi. Balki masofa 0 bo'lib qolishi ham mumkin! Bu esa simmetriya protsedurasi kerakli ishni bajarayotganini va biz uni chaqirishimiz mumkinligini bildiradi. Protседura ishlashi tugaganidan keyin chapga bir qadam yurish kerak bo'ladi, chunki Robotdan o'ng devorgacha bo'lgan masofa uning boshlang'ich joyidan chap devorgacha bo'lgan masofaga teng bo'lishi kerak. Mana nima hosil bo'ladi:

PROT simmetriya

BOSHLANISH

AGAR EMAS chap bo'sh

U HOLDA

TOKI o'ng bo'sh BAJAR

o'ngga

TAMOM

AKS HOLDA

chapga

simmetriya

chapga

TAMOM

TAMOM

Buni qarangki, yozilgan bu algoritm har qanday sharoitda ham to'g'ri ishlar ekan! Qo'yilgan masalani yechadigan algoritm esa juda sodda ko'rinishga ega:

simmetriya

Natijada bizda juda g'alati protsedura hosil bo'ldi: u o'zini o'zi chaqiradi. Bunday protseduralarni **rekursiv** deb atashadi. E'tibor bilan uning ishini qadamba-qadam o'rganib chiqamiz.

Robot bilan chap devor orasidagi boshlang'ich masofa 0 ga teng bo'lganda protsedura to'g'ri ishlashini ko'rdik. Robot bilan chap devor orasidagi masofa bitta katak bo'lganda qanday voqea ro'y berishini ko'rib chiqamiz.

Protседura boshida **EMAS chap bo'sh** sharti **YOLG'ON**, shuning uchun protseduradagi tarmoqlanish tuzilmasining **AKS HOLDA** so'zidan keyingi qismi bajariladi. Bu qismda uchta ko'rsatma bor:

**chapga
simmetriya
chapga**

Birinchi ko'rsatmani bajargach, Robot chap devorga bitta katak yaqinlashadi, ya'ni devor yoniga borib qoladi. Keyin **simmetriya** protsedurasi bajariladi. Endi vaziyat o'zgardi: Robot chap devor yonida turibdi. Bu vaziyatda protsedura to'g'ri ishlashini bilamiz va u o'ng devor yoniga boradi. Uchinchi **chapga** ko'rsatmasi uni o'ng devordan bitta chap katakka o'tkazadi, ya'ni kerakli katakka tushadi va protsedura ishi tugaydi.

Siz, albatta, Robot bilan chap devor orasidagi masofa ikkita katak bo'lganda protsedura ishining to'g'riligini qanday tekshirishni tushungan bo'lsangiz kerak. **EMAS chap bo'sh** sharti **YOLG'ON**, demak, protsedurada tarmoqlanish tuzilmaning **AKS HOLDA** so'zidan keyingi qismi bajariladi. Birinchi ko'rsatmani bajargach, Robot chap devorga bitta katak yaqinlashadi, ya'ni devor bilan uni orasida bitta katak qoladi. Keyin simmetriya protsedurasi bajariladi va Robotni, yuqorida ko'rib o'tganimizdek, o'ng devordan bitta chap katakka o'tkazadi. Keyingi chapga ko'rsatmasi uni o'ng devordan ikkita chap katakka, ya'ni kerakli katakka o'tkazadi.

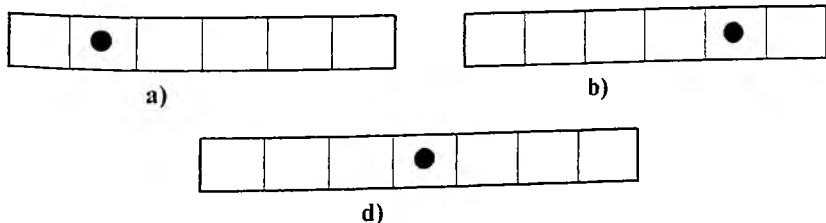
Mulohaza yuritishning bu usuli – **induksiya bo'yicha mulohaza** – bizga birinchi marta duch kelishi emas. U bizga chap devordan ixtiyoriy boshlang'ich holatda protsedura ishining to'g'riligini tekshirish imkonini beradi.

7.1-mashq

Nima uchun oxirgi **chapga** ko'rsatmasidan oldingi **simmetriya** protsedurasiga **chap bo'sh** shartini tekshirmasa ham bo'ladi?

7.2-mashq

Robotning 7.2-rasmdagi holatlari uchun **simmetriya** protsedurasi bajarilgandagi qadamlar ketma-ketligini yozib chiqing.



7.2-rasm.

7.2-masala

Robot devorlar bilan o'ralgan to'g'ri to'rtburchakning ichida turibdi. Robotni boshlang'ich holatidan to'g'ri to'rtburchak markaziga nisbatan simmetrik bo'lgan katakka o'tkazuvchi algoritm tuzing.

7.3-masala

Robot kengligi 1 ga teng gorizontol yo'lakda turibdi. Yo'lak chap tomondan devor bilan chegaralangan, o'ng tomon – cheksiz. Robotni shunday katakka o'tkazingki, bu katakdan chap devorgacha bo'lgan masofa boshlang'ish katakdan chap devorgacha bo'lgan masofadan ikki marta katta bo'lsin.

Yechim. Bu masalada Robot chap devor yonida turgan bo'lsa, nima qilish tushunarli: u o'z joyida qolishi kerak. Quyidagicha protsedura tuzamiz:

PROT ikkilangan sakrash

BOSHLANISH

AGAR EMAS chap bo'sh

U HOLDA

...

AKS HOLDA

...

TAMOM

TAMOM

Nuqtalar o'rniga qanday ko'rsatmalar qo'yilishi kerak? Birinchi bo'lib, devorgacha bo'lgan masofani kamaytirish, ya'ni vaziyatni soddalashtirish uchun chapga ko'rsatmasini bajarish kerakdir. Keyin yana ikkilangan sakrash protsedurasini chaqirish lozim. So'ngra – devorgacha bo'lgan masofa ikki marta katta bo'lishi uchun o'ngga ikki qadam tashlash kerak bo'ladi.

Demak, har bir chapga bir qadam uchun o'ngga ikki qadam yurish mos keladi.

7.3-mashq

1) **Ikkilangan sakrash** protsedurasini oxirigacha yozib chiqing.

2) **EMAS chap bo'sh** shartini **chap bo'sh** sharti bilan almashtirib, boshqa **ikkilangan sakrash** protsedurasi variantini yozing.

3) Agar Robotning chap devorgacha boshlang'ich masofasi quyidagicha bo'lsa, Robotni **ikkilangan sakrash** protsedurasini bajarishidagi qadamlari ketma-ketligini yozib chiqing:

a) ikkita katak;

b) uchta katak.

Rekursiyadan chiqish

Rekursiv protseduralarni har qanday Ijrochi uchun yozish mumkin. Masalan, Robotni kvadrat bo'ylab yurishi uchun rekursiv protsedura yozaylik.

7.4-masala

Tomoni 5 ga teng kvadratni chizish rekursiv protsedurasini yozing.

Javob.

PROT rekursiv kvadrat

BOSHLANISH

oldinga

oldinga

oldinga

oldinga

oldinga

o'ngga

rekursiv kvadrat

TAMOM

Bu protsedura qanday ishlashini ko'rib chiqamiz. Demak, **rekursiv kvadrat** protsedurasi chaqirilganda

oldinga

oldinga

oldinga

oldinga

oldinga

o'ngga

ko'rsatmalari bajariladi. Bunda Robot kvadratning tomoni bo'ylab yurib 90 gradusga buriladi. Keyin yana **rekursiv kvadrat** protsedurasi chaqiriladi. Robot kvadratning boshqa tomoni bo'ylab yurib o'ngga buriladi. Keyin yana **rekursiv kvadrat** protsedurasi chaqiriladi. Yana ikki tomon o'tilgach kvadrat bo'ylab yurish yakunlanadi. Lekin nimadir bo'ldi? Robot to'xtamasdan yana shu kvadrat bo'ylab yurishni davom ettirmoqda. Bu esa cheksiz davom etadi – algoritim siklga tushib qoldi.

Vaziyat yoqimsiz. Biz doimo siklga tushib qolmaslikka harakat qilar edik. Bu holda undan qutulishni chorasini bormikan? Taassufki, yo'q. Haqiqatan, rekursiv protseduraning chaqirilishi cheksiz davom etmasligi uchun protseduraga chaqirilish yuzaga kelmaydigan shart kiritish kerak. Lekin bu Robotda bunday shart yo'q. Robotning algoritmi uning holatidan qat'iy nazar, bir xilda bajarilaveradi.

Shu kabi, Dehqon, Suvchi, Chigirtka, Oshiruvchi, Zohid kabi Ijrochilarda shart yo'q. Bu Ijrochilar uchun rekursiv protseduraning qo'llanishi yoki siklga tushib qolinadi yoki keyingi ko'rsatmani bajarsih mumkin bo'lmay qolib INKOR yuzaga keladi. Shuning uchun

**sharti yo'q Ijrochilar uchun rekursiv
protseduralarni qo'llamang**

Bu qo'llanmada ko'rib chiqilgan Ijrochilardan faqat Kamaytiruvchi va Robotda shart bor. Demak, faqat shu ikki Ijrochi uchun rekursiv protseduralarni yozish ma'noga ega.

Bizning mulohazalarimizdan shunday xulosaga kelish mumkin:

**rekursiv protseduralarni yozganda rekursiv chaqirish yuzaga
kelmaydigan shart ko'rsatilishi kerak.**

Bu juda kerakli xulosa! Shuning uchun ham avvalgi bo'limda algoritm namunalari keltirganimizda, rekursiv protseduralarning shunday shartlarini va zaruriy amallarining tavsifini yozishdan boshlaganmiz.

7.5-masala

Quyidagi masalalarda rekursiv chaqirish yuzaga kelmaydigan shartlarni hamda bu holda qanday amallar bajarilishi zarurligini ko'rsating:

- a) Kamaytiruvchi tomonidan ixtiyoriy sondan 0 ni hosil qilish;
- b) Robot o'tish joyi bo'lgan gorizontall devor tagida turibdi. O'tish joyi chapda yoki o'ngdaligi noma'lum. O'tish joyini toping;
- d) Robot o'tish joyi bo'lgan gorizontall devor tagida turibdi. O'tish joyi Robotdan chapda. Robot undan o'tib boshlang'ich holati ustiga borib turishi kerak;
- e) Robotdan yuqorida bo'yalgan katak bor. Robot maydonida devorlar yo'q. Robotni bo'yalgan katakdan yuqoridagi shunday katakka olib kelingki, Robot bilan bo'yalgan katak orasidagi masofa Robotning boshlang'ich holatidagi katak bilan bo'yalgan katak orasidagi masofadan uch marta katta bo'lsin.

Qachon rekursiyasiz ishlash mumkin

Ko'rinishidan sodda ko'ringani bilan rekursiya bilan ishlash ancha murakkab. Haqiqatan, Dehqon, Suvchi, Chigirtka uchun birinchi algoritmlarimizni eslang. Ularning to'g'riligini tekshirish

juda oson edi. Biz Ijrochining har bir ko'rsatmadan keyingi holatini kuzatib turar va barcha ko'rsatmalar bajarilgandan keyin uning holati masala shartini qanoatlantirishiga ishonch hosil qilar edik.

Takrorlash tuzilmasini kiritilishi tekshirishni unchalik murakkablashtirmadi. Ijrochining harakati avvalgidek uning holatiga bog'liq emasdi. Yagona qo'shimcha qiyinchilik shundan iborat ediki, juda qisqa yozuvli algoritm bajarilishi natijasida Ijrochi juda ko'p amallarni bajarishi kerak bo'lardi.

Shart kiritilishi algoritmning to'g'riligini tekshirishda boshqacha aks etadi. Endi Ijrochining bajarishi kerak bo'lgan ko'rsatmalar ketma-ketligi Ijrochining holatiga bog'liq bo'ldi – ish boshida Kamaytiruvchi ekranida qanday son aks etib turardi, Robot maydonida devorlar qanday joylashgan, Robot maydonining qaysi kataklari bo'yalgan. Algoritmni oddiygina bajarilishi va to'g'ri natijani olinishi algoritm to'g'ri ekanligiga ishontira olmaydi. Algoritm biz tekshira olgan yoki tekshirishni xohlagan hollardagina emas, balki ixtiyoriy holda ham to'g'ri ishlashiga ishonch hosil qilishimiz kerak. Albatta, ishonch hosil qilishning eng yaxshi usuli – bu algoritm ishining to'g'riligini isbotlashdir.

Rekursiyani qo'llash yana tekshirish qiyinchiligini chuqurlashtiradi. Haqiqatan, biz endi qaysi ko'rsatma bajarilayotganini kuzatibgina qolmasdan, bu ko'rsatma qayerdan paydo bo'lganini ham bilishimiz kerak. Aytaylik, 7.1-masala yechimidagi **simmetriya** rekursiyasi ishini tekshirayotib, navbatdagi **chapga** ko'rsatmasi rekursiv protsedurani uchinchi yoki beshinchi chaqirishda paydo bo'lgani va hokazoni yodda saqlashimiz kerak. Yana agar rekursiv protseduralar algoritmida ko'p bo'lib, bir-birini chaqirsa-chi?

Tekshirish qiyinligi – bu rekursiyalardan foydalanishdan tiyilishni sabablaridan faqat birrasidir. Boshqa sabablar ham bor: rekursiyali dasturlar rekursiyasiz dasturlarga nisbatan kompyuterda bajarayotganda sekin ishlaydi va xotiradan **ko'p** joy talab qiladi; rekursiyali dasturlarni rekursiyasiz dasturlarga **almashtirayotganda** masalaga boshqa tomondan qarashga yordam beruvchi ajoyib fikrlar paydo bo'lishi mumkin. Shu sababli, masalaning rekursiv yechimini topgach, yana o'ylab ko'ring: rekursiyasiz ishlash mumkin emasmikan?

7.6-masala

Kamaytiruvchi uchun ixtiyoriy sonni eng kam qadamda 0 ga olib keluvchi rekursiv protsedura tuzing. Quyidagi sonlar uchun protsedurangiz bajarilayotgandagi Kamaytiruvchining ko'rsatmalar ketma-ketligini yozing:

- a) 5;
- b) 9;
- d) 14.

Kataklarni bo'yab, Robot bilan ishlaganda rekursiyadan qutulish mumkin. Bunda bo'yalgan kataklar xotira vazifasini o'taydi. Bu holda **bo'yalgan** sharti Robot bu katakdan o'tgan yoki o'tmaganini bilish uchun imkon beradi.

7.7-masala

Robot devordan yuqorida qandaydir masofa uzoqlikda turibdi. Quyidagi imkoniyatlarda Robotni devorgacha olib borib joyiga qaytarib olib keluvchi protsedura tuzing:

- a) rekursiyani ishlatish mumkin;
- b) bo'yash mumkin, rekursiyani ishlatish mumkin emas.

Bu masalalardan birortasi ham sizga qiyinchilik keltirib chiqarmaydi, deb umid qilamiz.

Oraliq protsedura orqali rekursiv chaqirish

Ba'zan protseduralarni yozganda uning o'zini o'zining ichida boshqa protsedura orqali chaqirgan qulay. To'g'ri to'rtburchakni bo'yash masalasining rekursiv yechimini qaraymiz.

7.8-masala

Robot devorlar bilan o'ralgan to'g'ri to'rtburchakning chap quyi burchagida turibdi. To'g'ri to'rtburchakning ichida devorlar yo'q. To'g'ri to'rtburchakning barcha kataklarini bo'yash algoritmini tuzing.

Yechim. Ikkita rekursiv protsedurani qo'llaydigan bo'yash algoritmini yozamiz:

PROT to'g'ri to'rtburchakni bo'ya

BOSHLANISH

AGAR EMAS bo'yalgan

U HOLDA

yo'lakni bo'ya va qayt

TAMOM

TAMOM

va

PROT yo'lakni bo'ya va qayt

BOSHLANISH

TOKI yuqori bo'sh BAJAR
bo'ya
yuqoriga

TAMOM

TOKI quyi bo'sh BAJAR
quyiga

TAMOM

AGAR o'ng bo'sh

U HOLDA

o'ngga

to'g'ri to'rtburchakni bo'ya

TAMOM

TAMOM

Endi algoritm bitta ko'rsatmadan iborat bo'ladi.

to'g'ri to'rtburchakni bo'ya

Bizning yechimimizda **to'g'ri to'rtburchakni bo'ya** protsedurasi **yo'lakni bo'ya va qayt** protsedurasini chaqiradi. O'z navbatida, **yo'lakni bo'ya va qayt** protsedurasi **to'g'ri to'rtburchakni bo'ya** protsedurasini chaqiradi. Shunday qilib, **to'g'ri to'rtburchakni bo'ya** protsedurasi o'zini-o'zi oraliq **yo'lakni bo'ya va qayt** protsedurasi orqali, **yo'lakni bo'ya va qayt** protsedurasi o'zini o'zi oraliq **to'g'ri to'rtburchakni bo'ya** protsedurasi orqali chaqiradi.

Bu holda ikkala protsedurani rekursiv deb hisoblaymiz va ular bilan oddiy protsedura kabi muomala qilamiz.

Nazorat savollari va topshiriqlar

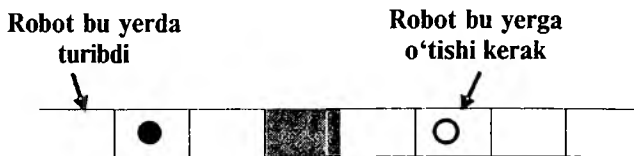
- 1. Qanday algoritmlar rekursiv deb ataladi?*
- 2. Simmetriya protsedurasini izohlab bering.*
- 3. Ikkilangan sakrash protsedurasini izohlab bering.*
- 4. Rekursiya bilan yuzaga keladigan muammolarni so'zlab bering.*
- 5. Bobdagi barcha mashqlarni bajaring.*

Qo'shimcha masalalar

R-1. Robot kengligi 1 katak bo'lgan gorizontal yo'lakda turibdi. Undan chapda qayerdadir bo'yalgan katak bor. Robotni turgan katagidan bo'yalgan katakka borib, yana joyiga qaytarib keltiradigan algoritm tuzing.

R-2. Robot kengligi 1 katak bo'lgan yuqoridan va quyidan devor bilan chegaralangan gorizontal yo'lakda turibdi. Undan o'ngda qayerdadir bo'yalgan katak bor. Robot turgan katagiga bo'yalgan katakka nisbatan

simmetrik bo'lgan chapdagi katakka borishi kerak (7.3-rasm). Mos algoritmni tuzing.



7.3-rasm.

R-3. Robot kengligi 1 katak bo'lgan gorizontaal yo'lakda turibdi. Undan chapda qayerdadir devor bor. Robotni turgan katagidan devorgacha olib borib yana joyiga qaytarib olib keladigan algoritm tuzing.

R-4. Robotni o'zidan yuqorida joylashgan gorizontaal devor yoniga olib keladigan rekursiv algoritm tuzing.

R-5. Robot o'tgan katagini bo'yab ketishi mumkin bo'lsa, u holda

a) R-1

b) R-3

masalalarni rekursiyasiz yeching.

K-1. Samarador Kamaytiruvchi uchun rekursiv protsedurani chaqiradigan algoritm tuzing.

VIII bob. SARALOVCHI UCHUN MASALALAR

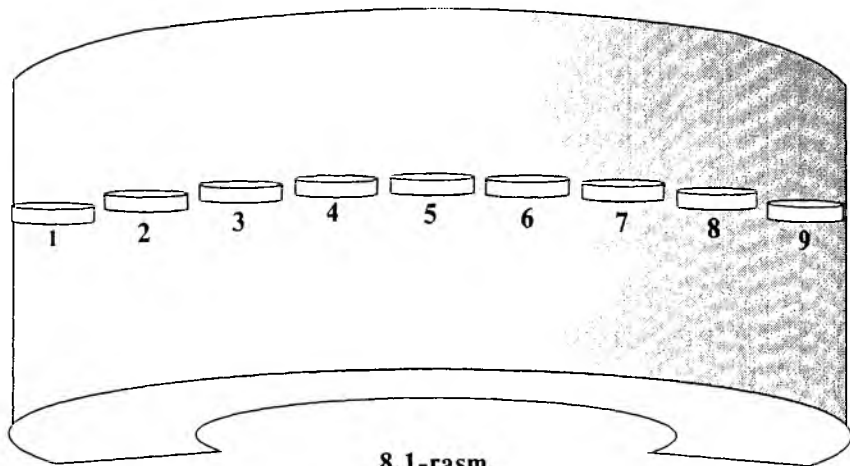
Saralovchi tarixi

Uch kishini bir necha yillardan buyon uzoq kosmosda olis sayyoralar tomon olib ketayotgan kosmik kema g'alati oltinrang bulut orasidan uchib o'tdi. Kemada tug'ilgan va bir yosh bo'lgan Bek ismli bola hali yurishni o'rganmagani uchun g'ildirakli aravachasida katta doira shaklidagi xonasida bir o'zi aylanib yurishni yoqtirardi. U aravachasini g'ildiratib borib o'yinchoqlarini egilib olar, o'ynab yana otib yuborar edi. Bek oltinrang bulut ta'sirida o'zida bir necha xil o'zgarishlarni sezdi. Birinchi o'zgarish zehni bilan bog'liq bo'lib, u endi bola bo'lsa ham arifmetik amallarni juda tez bajarar, har xil narsalarni turli ko'rinishdagi miqdorini bir qarashda juda tez aniqlab, taqqoslay olardi. Ikkinchisi, u tajanglashganligi va endi bolalar o'yinchoqlarini o'ynagisi kelmay qolganligi edi. Xonada bir o'zi zerikib qolganligidan chinqirib yig'lay boshladi. O'g'lini ovutishga kelgan ota-ona bolani qunt bilan ota-onasining suhbatini tinglayotganiga e'tibor berishdi. O'g'lidagi o'zgarishlarni sezgach, zehni o'tkir o'g'lini ovutish uchun ota-onasi xonani boshqacha jihozlab, faqat bir qo'li bor hamda g'ildirakda yuradigan sodda dasturli Saralovchi I nomli robot yasab berishdi. Keyin robotni boshqarishga mo'ljallangan dastur tuzishga yo'naltirilgan turli boshqotirmali masalalarni ayta boshlashdi. Biz endi Bek bilan birga shu masalalarni ko'rib chiqamiz.

Avval xona haqida. Aytganimizdek, doira shaklidagi devorlari bo'ylab bir xil balandlikda tokchalar o'rnatildi. Har bir tokchaga tartib raqami berildi, xonani bir qismining ko'rinishi 8.1-rasmda ifodalangan.

Kerak bo'lsa, tokchalarni bir nechtasini qoldirib, qolganlarini yopib qo'yish mumkin edi. Har bir tokchani qulaylik uchun quyidagicha nom bilan belgilab olamiz:

tokcha(1), tokcha(2), ..., tokcha(7), ..., tokcha(21), ..., tokcha(1963), ... va shu asosida ixtiyoriy A tokchadagi buyumni **tokcha(A)** orqali belgilay olamiz. Ya'ni, **tokcha(A)** deganda Bek va Saralovchi I tokchani emas, shu tartib raqamli tokchadagi



8.1-rasm.

buyumni tushunadi. Tokchalar sonining **quyi chegarasi** – INF, **yuqori chegarasi** – SUP kabi belgilanadi. Yuqoridagi belgilashlarga ko‘ra $INF=1$, SUP esa masala shartlarida aniqlab boriladi.

Avval, tokchalarni alifbo bo‘yicha A, B, D kabi belgilamoqchi bo‘ldik. Lekin tokchalar soni ko‘payganda harflar ikkitalab, uchtalab va shu kabi qo‘shib yozilishi anchagina qiyinchilik keltirib chiqarishi noqulaylikka sabab bo‘lardir. Shuning uchun tokchalar soni 10 tadan kam bo‘lsa, biz haflardan ham foydalanaveramiz. Ya‘ni, A harfi tokcha(1) ni, B harfi tokcha (2) ni va shu kabi tushunilaveradi. Bu ma‘lumotlarning hammasi Bek va Saralovchi I ga tushunarli ekan.

Yangi Ijrochimiz Saralovchi I sonlar ustida arifmetik amallarni bajara oladi va tekshirilayotgan quyidagi shartlarni **ROST** yoki **YOLG‘ON** ekanligini biladi, ya‘ni bu shartlarni **tekshira oladi** hamda shu asosida mantiqiy **xulosa** chiqara oladi:

$=$ (teng), $<>$ (teng emas), $<$ (kichik), $< =$ (katta emas), $>$ (katta), $> =$ (kichik emas).

Sodda dasturli Ijrochi Saralovchi I ning ko‘rsatmasi faqat bitta: **o‘tkaz tokcha(N), tokcha (M)**

Bu amalni bajarganda Saralovchi I qo‘li bilan **tokcha (N)** dagi buyumni ko‘tarib, **tokcha (M)** ga olib qo‘ya oladi. Agar bu jarayonda **tokcha(M)** bo‘sh bo‘lmasa, undagi buyumni surib tushirib yuborib, bo‘shatadi va so‘ng **tokcha (N)** dagi buyumni qo‘ya oladi. Umuman, buyum qo‘yilayotgan tokchani **bo‘sh** deb hisoblash mumkin, baribir undagi buyumlar tashlab yuboriladi. Bek faqat shu axborotlar asosida Saralovchi I ni boshqarish

dasturini tuzishi kerak. **Bu yerda** qiziq bir usulni ko'rish mumkin, miqdorni o'zgartirib **uning avvalgi** nomini saqlab qoldik. Bundan qiymati o'zgaradigan miqdor tushunchasi paydo bo'ladi.

O'tkazish masalalari

8.1-masala

Saralovchi I 1-tokchadagi buyumni 2-tokchaga olib qo'yishi kerak.

Javob. Bek faqat bittagina ko'rsatma yozdi:

o'tkaz tokcha(1), tokcha(2)

yoki

o'tkaz A, B

8.1-mashq

Saralovchi I N-tokchadagi buyumni M-tokchaga olib qo'yishi kerak.

Bekning ota-onasi **tokcha(2)** dagi buyum sinishi yoki boshqacha ahvolda yaroqsiz bo'lib qolishini xohlashmasa, buni alohida ta'kidlashligi kerakligini tushunib qolishdi.

8.2-masala

Saralovchi I 1-tokchadagi buyumni 2-tokchaga, 2-tokchadagi buyumga zarar yetkazmasdan o'tkazishi kerak.

Javob. Bek zehni bo'lgani uchun 2-tokchadagi buyumni 1-2-tokchadan boshqa ixtiyoriy yordamchi tokchaga, masalan, 3-tokchaga olib qo'yishni buyurgan, chunki masala shartida boshqa tokchalardagi buyumlarga zarar yetkazmaslik haqida hech narsa deyilmagan-da!

Algoritmida ikkitagina ko'rsatma yoziladi:

o'tkaz tokcha (2), tokcha (3)

o'tkaz tokcha (1), tokcha (2)

yoki

o'tkaz B, D

o'tkaz A, B

8.2-mashq

Saralovchi I 5-tokchadagi buyumni 9-tokchaga, 9-tokchadagi buyumga zarar yetkazmasdan olib qo'yishi kerak.

8.3-masala

Saralovchi I tartib raqami 100 dan katta bo'lmagan N-tokchadagi buyumning tartib raqami 100 dan katta bo'lmagan

M-tokchaga, M-tokchadagi buyumga zarar yetkazmasdan olib qo'yishi kerak.

Javob. Bek bu holda salgina o'ylanib qoldi. Chunki N va M ning qiymatini, ya'ni qaysi tokchalar haqida gap borayotganini ota-onasi hech narsa aytishmadiku! Faqatgina $M \leq 100$ va $N \leq 100$ ekani ma'lum. Bu holda ham zeharli Bekning javobi ancha tez bo'ldi:

o'tkaz tokcha(M), tokcha(N+M)

o'tkaz tokcha(N), tokcha(M)

1-sharh

*Bu holda Bek $N-M$ yoki $M-N$ tartib raqamli tokchalarni ishlata olmas edi, chunki, masalan, $N = 2$ va $M = 1$ bo'lsa, u holda $N-M = 1 = M$ yoki $M-N < 1$, ya'ni **INKOR** holati yuzuga keladi.*

8.3-mashq

Shundan keyin uning ota-onasi ancha o'ylanib qolishdi. Nima haqida deb o'ylaysiz?

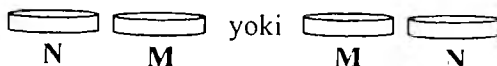
8.4-masala

Berilgan SUP ga ko'ra Saralovchi I N-tokchadagi buyumni M-tokchaga M-tokchadagi buyumga zarar yetkazmasdan olib qo'yishi kerak, bunda algoritmdagi tokchalarni tartib raqami SUP dan katta bo'lishi mumkin emas.

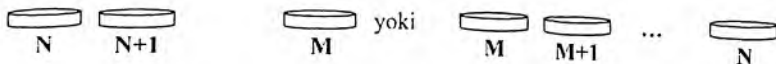
Yechim. Masalada chegara berilgan, ya'ni avvalgi masalaning tahlilidan $N+M \leq \text{SUP}$ bo'lishi nazarda tutilmoqda, shekilli, mana Bekning ota-onasi nima haqida o'ylashgan ekan!

Bek bu holda ancha o'ylanib qoldi. Chunki, $N+M$ ning qiymati tokchalar sonidan katta, ya'ni $N+M > \text{SUP}$ bo'lib qolishi mumkin. Lekin tokchalar soni 3 tadan kam emas, aks holda yechim yo'q.

Keling, bola o'ylab topgunicha, biz unga yordamlashib yuboramiz. Masala shartida ikkita tartib raqamli tokchadagi buyumlarga zarar yetkazmaslik haqida aytilgan. Demak, biz boshqa birorta yordamchi tokchani aniqlashimiz kerak. Saralovchi I da shart tekshirish imkoniyati borligi va tokchalar soni ikkitadan ortiqligi esingizda bo'lsa kerak. Ikki holni ajratamiz: tokcha tartib raqamlari teng bo'lmagani uchun tokchalar yoki yonma-yon turadi (8.2-rasm) yoki ular orasida hech bo'lmaganda bitta tokcha (8.3-rasm) bo'ladi.



8.2-rasm.



8.3-rasm.

2-sharh

Bu masalada ham Bek N-M yoki M-N tartib raqamli tokchalarni ishlata olmaydi.

Barcha hollarni ko'rib chiqamiz.

1-hol. $M = N+1$ bo'lsin, u holda $M > N$. Demak, yordamchi tokcha $N-1$ yoki $M+1$ bo'ladi. Bu tokchalardan qaysi biri ekanligini qanday aniqlaymiz? Agar $N > 1$ bo'lsa, javob $N-1$, aks holda, uchinchi tokcha mavjudligini hisobga olsak, javob $M+1$.

2-hol. $N = M+1$ bo'lsin, u holda $N > M$. Demak, yordamchi tokcha, agar $M > 1$ bo'lsa: $M-1$, aks holda, uchinchi tokcha mavjudligidan: $N+1$.

Agar $M = N+1$, $M-N=1$, $N-M= -1$ bir xil shartligini va $N- = M+1$, $N-M = 1$, $M-N = -1$ bir xil shartligini e'tiborga olsak, ko'rinib turibdiki, yonma-yon turish sharti quyidagicha ekan:

$$(M-N) \cdot (N-M) = -1.$$

Cho'zib o'tirmasdan algoritmnı yozishni boshlaymiz:

AGAR $(M-N) \cdot (N-M) = -1$ {bu holda N va M yonma-yon}

U HOLDA

AGAR $M-N=1$ {bu holda $M=N+1$, ya'ni $M > N$ }

U HOLDA

AGAR $N > 1$

U HOLDA

o'tkaz tokcha(M), tokcha(N-1)

o'tkaz tokcha(N), tokcha(M)

AKS HOLDA

o'tkaz tokcha(M), tokcha(M+1)

o'tkaz tokcha(N), tokcha(M)

TAMOM

AKS HOLDA {bu holda $N=M+1$, ya'ni $N > M$ }

AGAR $M > 1$

U HOLDA

o'tkaz tokcha(M), tokcha(M-1)

o'tkaz tokcha(N), tokcha(M)

AKS HOLDA

o'tkaz tokcha(M), tokcha(N+1)

o'tkaz tokcha(N), tokcha(M)

TAMOM
TAMOM
AKS HOLDA

{ N va M yonma-yon emas}

...

TAMOM

Juda ham uzun-a! Yaxshi ham o'ngga surish orqali algoritm qismlarini ajratib oldik.

3-sharh

Ba'zan algoritmda { va } orasida tushunish oson bo'lishi uchun izohlar berib boramiz.

3-hol. Tokchalar yonma-yon emas, ya'ni $N <> M+1$ va $M <> N+1$ yoki $(M-N) \cdot (N-M) <> -1$. Bu holda N -tokcha va M -tokcha orasida hech bo'lmaganda bitta tokcha bor. Ma'lumki, $M+N$ va $N+M+1$ sonlardan biri juft bo'ladi. U holda sonlarning o'rta arifmetigi haqidagi xossalarni va $[a]$ a sonining butun qismi ekanligini inobatga olsak, quyidagi tengsizlikdan biri o'rinli:

$$N < [(N+M+1):2] < M \text{ yoki } M < [(N+M+1):2] < N.$$

Demak, bu holda yordamchi tokcha $[(N+M+1):2]$ bo'lar ekan. Endi nuqtalar o'rniga quyidagini yozish yetarli:

o'tkaz tokcha(M), tokcha([(N+M+1):2])

o'tkaz tokcha(N), tokcha(M)

8.4-mashq

Sonning butun qismi amalidan foydalanmasdan 3-holni hal eting.

Buni qarang, Siz masalaning yechimini o'rganib chiqquncha Bek ham masalani quyidagicha hal qilib qo'yibdiya, qoyil!

AGAR $N < M$ {bu holda $N < M$ }

U HOLDA

AGAR $1 < N$ {bu holda $1 < N < M$ }

U HOLDA $1 < N < M$, ya'ni $N-1$ -tokcha bo'sh}

o'tkaz tokcha(M), tokcha(N-1)

o'tkaz tokcha(N), tokcha(M)

AKS HOLDA {bu holda $N = 1 < M$ }

AGAR $N = M-1$ {bu holda, $N = 1$ va $M=2$ }

U HOLDA

o'tkaz tokcha(M), tokcha(M+1) { $M+1=3$ bo'sh}

o'tkaz tokcha(N), tokcha(M)

AKS HOLDA

{ $N=1$ va $M>2$, ya'ni $(M-1)$ -tokcha bo'sh}
o'tkaz tokcha(M), tokcha(M-1)
o'tkaz tokcha(N), tokcha(M)

TAMOM

TAMOM

AKS HOLDA {bu holda $M<N$, chunki $M=N$ bo'lolmaydi}

AGAR $1<M$ {bu holda $1<M<N$ }

U HOLDA { $1<M<N$, ya'ni $M-1$ -tokcha bo'sh}

o'tkaz tokcha(M), tokcha(M-1)

o'tkaz tokcha(N), tokcha(M)

AKS HOLDA {bu holda $M=1<N$ }

AGAR $M=N-1$ {bu holda, $M=1$ va $N=2$ }

U HOLDA

o'tkaz tokcha(M), tokcha(N+1) { $N+1=3$ bo'sh}

o'tkaz tokcha(N), tokcha(M)

AKS HOLDA

{ $M=1$ va $N>2$, ya'ni $(N-1)$ -tokcha bo'sh}

o'tkaz tokcha(M), tokcha(N-1)

o'tkaz tokcha(N), tokcha(M)

TAMOM

TAMOM

TAMOM

Bu yechim ham uzun bo'lgani bilan uning e'tiborli tomonlaridan biri, Bek sonni butun qismi kabi Saralovchi I hozircha bilmaydigan amalni qo'llamagan. Ikkinchisi, tashqi AGAR tuzilmasidagi U HOLDA va AKS HOLDA qismlariga e'tibor bersangiz, mantig'i juda ham o'xshash, yozuvlarda deyarli simmetriya bor. Mana mantiqiy fikrlashning samarasi, Bek bizga nisbatan zehnlilik chiqib qoldi-yov!

8.5-masala

Saralovchi I 1-tokchadagi buyum bilan 2-tokchadagi buyumlar o'rnini almashtirishi kerak.

Yechim. Bek uchun bu muammo emas ekan:

o'tkaz tokcha(2), tokcha(3)

o'tkaz tokcha(1), tokcha(2)

o'tkaz tokcha(3), tokcha(1)

yoki

o'tkaz B, D

o'tkaz A, B

o'tkaz D, A

Barakalla, Bek!

Tanlash masalalari

Farzandlarining aqlliligidan mehri jo'shib ketgan Bekning otanasi robotga ikkinchi qo'l o'rniga bo'sh tokcha vazifasini o'tay oladigan **Zt** tokcha (zahira tokcha) o'rnatib berishdi. Bu holda, birinchidan, robot **Zt** tokchada ixtiyoriy tokchadagi buyumning asl nusxasini hosil qila olardi, ya'ni tokchadagi buyumni xuddi o'ziga o'xshash boshqasini zaxiradan olib kela olardi, bunda **Zt** tokchadagi avvalgi nusxa tashlab yuboriladi. Ikkinchidan, **Zt** tokchadagi nusxani asosiy tokchalarga ham o'tkazish mumkin, bu holda ham robot zaxiradan foydalanadi. Yuqoridagi izohlarni hisobga olib, yangi Ijrochi Saralovchi II uchun quyidagi ko'rsatmani izohlaymiz:

o'tkaz tokcha(N), Zt

Bu ko'rsatmada Saralovchi II N-tokchadagi buyumni **Zt** tokchaga olmaydi, faqat **Zt** tokchaga N-tokchadagi buyumning nusxasinigina oladi.

Shart tekshirish quyidagi talabga bo'ysunadi: Saralovchi II **Zt** tokchadagi buyum miqdorini hamda qo'lga olingan biror tokchadagi buyum miqdorini aniqlaydi, keyin bir-biri bilan taqqoslaydi. Bu esa Saralovchi II

AGAR <shart>

U HOLDA

<ko'rsatmalar guruhi>

TAMOM

yoki

AGAR <shart>

U HOLDA

<1-ko'rsatmalar guruhi>

AKS HOLDA

<2-ko'rsatmalar guruhi>

TAMOM

kabi shartli tuzilmalarni bajara oladi, degani.

Avvalgi boblarda birikkan shartlar haqida ma'lumotlar berilgan edi. Saralovchi II robot ham bu ishni bajara oladi. Masalan,

tokcha(1)=0 VA Zt>0

sharti talabga qarab 1-tokchadagi buyum miqdori 0 ga tengligini va **Zt** tokchadagi buyumning miqdori 0 dan kattaligini va rostlik jadvali asosida tekshiradi. Saralovchi II da faqat bitta qo'l va **Zt** tokcha bor bo'lgani uchun ikkita shartdan ko'pi bilan ishlay olmaydi.

8.6-masala

Xonada 5 ta tokcha bo'lib, ularning ustiga turli sondagi bir xil o'lchamdagi kublar ustma-ust taxlangan. Saralovchi II bitta tokchada eng ko'p taxlangan kublar sonini aniqlashi kerak.

Yechim. Saralovchi II eng ko'p taxlangan kublar sonini aniqlashi uchun Zt tokchaga shu kublarning nusxasini olishi yetarli. Buning uchun Bek quyidagicha yo'l tutgan.

Birinchi qadamda Zt tokchaga 1-tokchadagi buyum nusxasi olindi. Demak, u hozircha eng ko'p kublar soni 1-tokchadagi kublar soniga teng, deb oldi. Saralovchi II Zt tokchadagi kublar bilan qo'lga olingan 2-tokchadagi kublar sonini taqqoslaydi. Agar 2-tokchadagi kublar soni ko'p bo'lsa, Zt tokchaga 2-tokchadagi kublarning nusxasini oladi, aks holda 3-tokchaga o'tadi. Demak, Bekning algoritmi quyidagicha ekan:

o'tkaz tokcha(1), Zt

AGAR Zt<tokcha(2)

U HOLDA

o'tkaz tokcha(2), Zt

TAMOM

AGAR Zt<tokcha(3)

U HOLDA

o'tkaz tokcha(3), Zt

TAMOM

AGAR Zt<tokcha(4)

U HOLDA

o'tkaz tokcha(4), Zt

TAMOM

AGAR Zt<tokcha(5)

U HOLDA

o'tkaz tokcha(5), Zt

TAMOM

Barakalla, Bek muammoni hal qildi. Lekin keyingi masalada nima qilar ekan?

8.7-masala

Xonada 1963 ta tokcha bo'lib, ularning ustiga turli sondagi bir xil o'lchamdagi kublar ustma-ust taxlangan. Saralovchi II bitta tokchada eng ko'p taxlangan kublar sonini aniqlashi kerak.

Yechim. Bek har bir tokchaga mos taqqoslashlarni erinmasdan juda ko'p vaqt sarflab Saralovchi II xotirasiga yozib kiritdi. Uning bu erinmasdan ishlagani uchun mukofot tarzida ota-onasi

Saralovchi II dasturiga TAKRORLANSIN – MARTA tuzilmasini va quyidagi qoidani kiritishdi.

Sintaksis qoidalari:

- N, Zt, tokcha(N) nom bo'lgani uchun bosh harflarda ham kichik harflarda ham yozilishi mumkin;
- TAKRORLANSIN k MARTA tuzilmasida **tokcha(i)** yozuvi takrorlanishda 1 dan k gacha sanalganda har bir songa mos tokchani anglatadi, ya'ni sanoq 1 bo'lsa – tokcha(1) ni, sanoq 2 da – tokcha(2) ni, ... , sanoq k da - tokcha(k) ni qaralayot-ganini bildiradi.

Endi 8.7-masala yechimi Saralovchi II uchun quyidagicha bo'ladi:

o'tkaz tokcha(1), Zt
TAKRORLANSIN 1963 MARTA
AGAR Zt<tokcha(i)
U HOLDA
o'tkaz tokcha(i), Zt
TAMOM
TAMOM

Bu algoritm quyidagicha ishlaydi:

Birinchi qadamda Zt tokchaga 1-tokchadagi buyum nusxasi olinadi. Demak, hozircha eng ko'p kublar soni 1-tokchadagi kublar soniga teng. Keyingi qadamda takrorlanish tuzilmasi ishlay boshlaydi. Sanoq 1 bo'lganda Saralovchi II Zt tokchadagi kublar bilan 1-tokchadagi kublar sonini taqqoslaydi, Zt tokchada 1-tokchadagi buyum nusxasi bo'lgani uchun shart bajarilmaydi. Sanoq 2 bo'lganda Saralovchi II Zt tokchadagi kublar bilan 2-tokchadagi kublar sonini taqqoslaydi. Agar 2-tokchadagi kublar soni ko'p bo'lsa Zt tokchaga 2-tokchadagi kublarning nusxasini oladi va takrorlanish qadami oshadi aks holda faqat takrorlanish qadami oshadi, va hokazo.

4-sharh

E'tibor bergan bo'lsangiz, TAKRORLANSIN – MARTA tuzilmasining bu ko'rinishida 1-tokchadagi buyum o'zining nusxasi bilan taqqoslandi. Bu tuzilmaning kamchiliklaridan biridir.

8.8-masala

Xonada 21 ta tokcha bo'lib, birinchi 7 tasi bo'sh, qolganlarining ustiga turli sondagi bir xil o'lchamdagi kublar ustma-ust taxlangan.

Saralovchi II bitta tokchada eng ko'p taxlangan kublar sonini aniqlashi kerak.

Bek bu masalaga ham avvalgi masaladagi yechimni yozdi, faqat 1963 o'rniga 21 oldi. Saralovchi II algoritmi bema'lol bajardi va har qanday boshlang'ich qiymatlarda faqat to'g'ri natijalar ko'rsatdi.

8.5-mashq

Algoritm haqiqatan masala yechimini berishiga ishonch hosil qiling.

8.9-masala

Xonada 21 ta tokcha bo'lib, birinchi 7 tasi bo'sh, qolganlarining ustiga turli sondagi bir xil o'lchamdagi kublar ustma-ust taxlangan. Saralovchi II kubi bor tokchalardan bitta tokchada eng kam taxlangan kublar sonini aniqlashi kerak.

Yechim. Bek tuzgan quyidagi algoritmi xato natijalar berdi:

o'tkaz tokcha(1), Zt

TAKRORLANSIN 21 MARTA

AGAR Zt>tokcha(i)

U HOLDA

o'tkaz tokcha(i), Zt

TAMOM

TAMOM

Xato nimadan iborat edi? 1-tokchada Zt da kub yo'q, ya'ni ularning soni 0 ga teng. 2-tokchadan 7-tokchagacha shunday natijaga ega bo'laveringiz. Lekin keyingi tokchalarda kublar soni 0 dan katta bo'lishiga qaramay yana 0 natijaga ega bo'laveringiz. Chunki sizga ma'lum, 0 va ixtiyoriy natural sondan kichigi yana 0 bo'ladi.

Buni tushungan Bek birinchi 7 ta tokchani tashlab yuborishga harakat qildi va quyidagicha algoritmi tuzdi:

o'tkaz tokcha(8), Zt

TAKRORLANSIN 21 MARTA

AGAR Zt>tokcha(i)

U HOLDA

o'tkaz tokcha(i), Zt

TAMOM

TAMOM

Yana o'sha ahvol, barcha boshlang'ich qiymatlarda faqat 0 natija olindi. Nima uchun? Chunki, 1-qadamda Zt tokchaga 8-

tokchadagi kublar nusxasi ko'chiriladi, ularning soni, albatta, shartga ko'ra 0 dan katta. 2-qadamda takrorlanish tuzilmasi ishga tushadi va sanoq birdan boshlanadi. 1-tokchada esa kub yo'q, natija yana 0 ta bo'ladi, avvalgi algoritm ishiga qaytib qolindi.

5-sharh

E'tibor bergan bo'lsangiz, bu Bekning emas TAKRORLANSIN – MARTA tuzilmasining kamchiligidir.

O'g'lining qiynalayotganini ko'rgan ota-ona yangi universal tuzilmani taklif etishdi. Tuzilma va uning sintaksis qoidalarini Saralovchi II ning dasturiga kiritishdi. Tuzilmaning umumiy ko'rinishi quyidagicha:

k DAN h GACHA BAJAR

<takrorlanishi lozim bo'lgan ko'rsatmalar>

TAMOM

Bu tuzilmada sanoq k dan boshlanadi va toki sanoq h ga yetguncha bittadan oshirib boriladi. Har qadamda BAJAR va TAMOM orasidagi takrorlanishi lozim bo'lgan ko'rsatmalar bajariladi. Endi sharhlarda ifodalangan kamchiliklarni yo'qotish mumkin. 8.9-masalaning yechimi esa quyidagicha:

o'tkaz tokcha(8), Zt

9 DAN 21 GACHA BAJAR

AGAR $Zt > \text{tokcha}(i)$

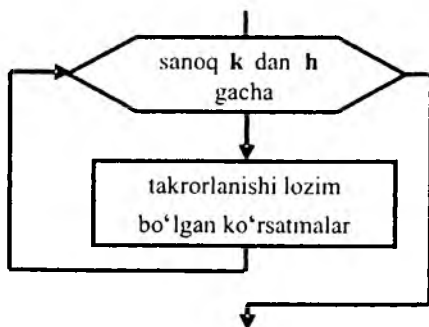
U HOLDA

o'tkaz tokcha(i), Zt

TAMOM

TAMOM

Yangi universal tuzilmaning blok-sxemasi quyidagicha:



8.4-rasm.

TAKRORLANSIN <son> MARTA

<takrorlanishi lozim bo'lgan ko'rsatmalar>

TAMOM

tuzilmasi yangi tuzilma orqali quyidagicha almashtirilishi mumkin:

I DAN <son> GACHA BAJAR

<takrorlanishi lozim bo'lgan ko'rsatmalar>

TAMOM

8.9-masalani Saralovchi II tushunadigan birikkan shartlardan foydalanib hal etish mumkin edi. Lekin Bek hali birikkan shartlarni bilmagani uchun qo'llay olmasdi.

Ota-onasi birikkan shartlarni Bekka tushuntirgunicha biz yechimni yozib qo'yamiz:

o'tkaz tokcha(8), Zt

TAKRORLANSIN 21 MARTA

AGAR tokcha(i)>0 VA Zt>tokcha(i)

U HOLDA

o'tkaz tokcha(i), Zt

TAMOM

TAMOM

8.6-mashq

Algoritm haqiqatan masala yechimini berishiga ishonch hosil qiling.

8.10-masala

Xonada N ta tokcha bo'lib, ulardan M tasi bo'sh, qolganlarining ustiga turli sondagi bir xil o'lchamdagi kublar ustma-ust taxlangan. Saralovchi II kubli tokchalardan bitta tokchada eng kam taxlangan kublar sonini aniqlashi kerak.

Yechim. Bek shunday o'ylamoqda:

1) masala shartidan $M < N$ ekanini bilaman, ya'ni kamida bitta kubli tokcha mavjud;

2) agar Zt tokchaga kubi yo'q tokchani nusxasini o'tkzamsam, javob 0 ga teng chiqadi;

3) avval Zt tokchaga kamida bitta kubli tokchadagi kubni o'tkazishim kerak.

Demak, birinchi navbatda kamida bitta kubli tokcha topiladi va Zt tokchaga o'tkaziladi:

TAKRORLANSIN N MARTA

AGAR tokcha(i)>0

U HOLDA

o'tkaz tokcha(i), Zt

TAMOM

TAMOM

Bu algoritm natijasida kamida bitta kubi bo'lgan tokchalar nusxasi Zt tokchaga o'tkazilaveradi va oxirida kamida bitta kubi bo'lgan eng oxirgi tokcha nusxasi qoladi. Endi masala yechimi to'liq bo'lishi uchun 8.9-masalaning birikkan shartli yechimidagi takrorlanish sonini 21 dan N ga almashtirib, yuqoridagi algoritm davomiga qo'shish yetarli.

8.11-masala

Xonada N ta tokcha bo'lib, ulardan M tasi bo'sh, qolganlarining ustiga turli sondagi bir xil o'lchamdagi kublar ustma-ust taxlangan. Saralovchi II kubli tokchalardan bitta tokchada eng ko'p taxlangan kublar sonini aniqlashi kerak.

8.7-mashq

Algoritmni mustaqil tuzing.

Yangi Saralovchi sari

8.12-masala

Xonada N ta tokcha bo'lib, ular ustiga turli sondagi bir xil o'lchamdagi kublar ustma-ust taxlangan. Ba'zi tokchalar bo'sh bo'lishi ham mumkin. Saralovchi II eng ko'p kub taxlangan tokchadagi kublarni 1-tokchaga o'tkazishi kerak.

Yechim. Bek avval eng ko'p kub taxlangan tokchadagi kublar sonini aniqladi. Buning uchun u 8.11-masala yechimini yozdi:

o'tkaz tokcha(1), Zt

TAKRORLANSIN N MARTA

AGAR Zt < tokcha(i)

U HOLDA

o'tkaz tokcha(i), Zt

TAMOM

TAMOM

Natijada Saralovchi II ning Zt tokchasida kerakli tokchani nusxasi hosil bo'ldi. Eng ko'p kub taxlangan tokchalar bir nechta bo'lishi mumkin.

Buni va 1-tokchadagi kublar tushirib yuborilishi Bek uchun ahamiyati yo'q, chunki bittasining nusxasini 1-tokchaga ko'chirish

masala hal bo'lishi uchun yetarli. Algoritm to'liq bo'lishi uchun u quyidagi ko'rsatmani qo'shib qo'ydi:

o'tkaz Zt, tokcha(1)

8.13-masala

Xonada N ta tokcha bo'lib, ular ustiga turli sondagi bir xil o'lchamdagi kublar ustma-ust taxlangan. Ba'zi tokchalar bo'sh bo'lishi ham mumkin. Tokchalardan bittasida taxlangan kublar soni eng ko'p. Saralovchi II 1-tokchadagi kublar bilan eng ko'p kub taxlangan tokchadagi kublarning o'rmini almashtirishi kerak.

Yechim. Bek avval eng ko'p kub taxlangan tokchadagi kublar sonini aniqladi:

o'tkaz tokcha(1), Zt

TAKRORLANSIN N MARTA

AGAR Zt < tokcha(i)

U HOLDA

o'tkaz tokcha(i), Zt

TAMOM

TAMOM

Endi Zt tokchada eng ko'p kub taxlangan tokchadagi kublar nusxasi bor. Yechim to'liq bo'lishi uchun quyidagini qo'shish mumkin:

TAKRORLANSIN N MARTA

AGAR Zt = tokcha(i)

U HOLDA

o'tkaz tokcha(1), tokcha(i)

TAMOM

TAMOM

o'tkaz Zt, tokcha(1)

1-tokchaga o'tkazilishi kerak bo'lgan tokcha bitta bo'lgani uchun U HOLDA va TAMOM orasidagi ko'rsatma takrorlanish nechta bo'lishidan qat'iy nazar faqat bir marta ishlaydi.

6-sharh

E'tibor qilgan bo'lsangiz, hozirgacha Zt tokcha Saralovchi uchun xotira bo'lib hizmat qilmoqda.

8.8-mashq

Xonada N ta tokcha bo'lib, ular ustiga turli sondagi bir xil o'lchamdagi kublar ustma-ust taxlangan. Ba'zi tokchalar bo'sh bo'lishi ham mumkin. Saralovchi II 1-tokchadagi kublar bilan

eng ko'p kub taxlangan tokchadagi kublarning o'rnini almash-tirishi kerak. Bu masalaga yuqoridagi algoritm javob bera oladimi?

Bekning muvaffaqiyatlaridan quvongan ota-onasi Saralovchi II ga qo'shimcha vazifalar kiritib yangi Ijrochi **Saralovchi III** ni hosil qilishdi. Bolalalik chog'laringizda zehningizni sinab ko'rish uchun teskaricha sanashni so'rashgani, ya'ni, masalan, 10 dan 1 gacha, yodingizdami? Endi, birinchidan, Saralovchi takrorlash tuzilmasida teskari sanashni biladigan bo'ldi. Ikkinchidan, sizga tanish bo'lgan TOKI – BAJAR tuzilmasi bo'ldi. Uchinchisi, robotga o'rnatilgan Ek nomli ekrancha bo'lib, Ek da Zt tokchadagi nusxaga mos tokchaning tartib raqami aks etib turar edi. Endi Bek bermalol bu Ek xotiradan quyidagicha foydalanishi mumkin bo'ldi: agar Zt tokchaga 22-tokchaning nusxasi olingan bo'lsa, u holda tokcha(Ek) yozuvi tokcha(22) bilan bir xildir. Bek uchun bu quvonchli voqea edi. Chunki, masalan, 9.13-masala bu imkoniyatlar yordamida ham oson, ham samarali hal etiladi:

b'tkaz tokcha(1), Zt

TAKRORLANSIN N MARTA

AGAR Zt<tokcha(i)

U HOLDA

o'tkaz tokcha(i), Zt

TAMOM

TAMOM

o'tkaz tokcha(1), tokcha(Ek)

o'tkaz Zt, tokcha(1)

Ko'rdingizmi, endi yangi imkoniyatlar sababli N marta taqqoslashning keragi yo'q. Demak, qanchalik ko'p axborotga ega bo'lsang, ishing shunchalik unumli bo'lar ekan!

Ichma-ich joylashgan sikllar

Ba'zan takrorlanish tuzilmalari ichma-ich joylashtilishi mumkin. Masalan, ekranida 0 soni aks etib turgan Oshiruvchi uchun quyidagi tuzilmani qaraylik:

TAKRORLANSIN 2 MARTA

1 ni qo'sh

TAKRORLANSIN 3 MARTA

2 ga ko'paytir

TAMOM TAMOM

Bu tuzilma quyidagicha ishlaydi: birinchi qadamda tashqi TAKRORLANSIN 2 MARTA tuzilmasining birinchi qadami boshlanadi va ekrandagi 0 ga 1 qo'shiladi; ikkinchi qadamda TAKRORLANSIN 3 MARTA tuzilmasining birinchi qadami boshlanadi natija 1 ni 2 ga ko'paytiradi; uchinchi qadamda TAKRORLANSIN 3 MARTA tuzilmasining ikkinchi qadami boshlanadi va natija 2 ni 2 ga ko'paytiradi; to'rtinchi qadamda TAKRORLANSIN 3 MARTA tuzilmasining uchinchi qadami boshlanadi va natija 4 ni 2 ga ko'paytiradi, bu bilan ichki sikl ishini tugatadi; beshinchi qadamda tashqi TAKRORLANSIN 2 MARTA tuzilmasining ikkinchi qadami boshlanadi va natija 8 ga 1 ni qo'shadi; oltinchi qadamda TAKRORLANSIN 3 MARTA tuzilmasining birinchi qadami boshlanadi va natija 9 ni 2 ga ko'paytiradi; yettinchi qadamda TAKRORLANSIN 3 MARTA tuzilmasining ikkinchi qadami boshlanadi va natija 18 ni 2 ga ko'paytiradi; sakkizinchi qadamda TAKRORLANSIN 3 MARTA tuzilmasining uchinchi qadami boshlanadi va natija 36 ni 2 ga ko'paytiradi, bu bilan ichki sikl ishini tugatadi; lekin tashqi sikl ham 2 marta takrorlangani uchun u ham ishini tugatadi va ekranda 72 soni aks etadi.

Demak, tashqi siklning 1-qadamida ichki siklning 1-, 2-, 3-qadami, tashqi siklning 2-qadamida ham ichki siklning yana 1-, 2-, 3-qadami takrorlanadi.

Ichma-ich joylashgan universal tuzilmani ko'raylik:

1 DAN 4 GACHA BAJAR

2 DAN 3 GACHA BAJAR

TAMOM

TAMOM

Bu tuzilmada ham tashqi sikl va ichki sikllar quyidagicha bajariladi:

Tashqi sikl sanog'i	Ichki sikl sanog'i
1	2
	3
2	2
	3
4	2
	3
4	2
	3

Agar sikllar ichida tokchalarning tartib raqamini ko'rsatmoqchi bo'lsak, u holda tashqi sikl sanog'i uchun bir harfni, ichki sikl sanog'i uchun boshqa harfni olishimiz kerak bo'ladi. Chunki, sanoq vaqtida har safar 1 qo'shib boriladi. Agar tashqi va ichki sikllar uchun belgilash bir xil qilib **k** deb olinsa, quyidagicha xatolik ro'y beradi:

Tashqi sikl sanog'i k harfi orqali	Ichki sikl sanog'i k harfi orqali
k = 1	k = 2
	k = 3
k = 4	k = 2
	k = 3
k = 4	k = 2
	k = 3
...	...

Bu holda tashqi sikl sanoqdagi **k** ni 1 dan boshlaydi; ichki sikl sanoqdagi **k** ni 2 dan boshlaydi; ichki sikl sanoqdagi **k** ga 1 qo'shib 3 deb oladi; tashqi sikl sanoqdagi **k** ga 1 qo'shib 4 deb oladi; ichki sikl sanoqdagi **k** ni 2 dan boshlaydi; ichki sikl sanoqdagi **k** ga 1 qo'shib 3 deb oladi; tashqi sikl sanoqdagi **k** ga 1 qo'shib 4 deb oladi; ichki sikl sanoqdagi **k** ni 2 dan boshlaydi va shu tariqa takrorlanish to'xtamaydi.

Boshqa misol:

1 DAN 3 GACHA BAJAR

2 DAN 4 GACHA BAJAR

TAMOM

TAMOM

uchun esa tashqi sikl kerakli sanoqni bajarmay tugaydi:

Tashqi sikl sanog'i k harfi orqali	Ichki sikl sanog'i k harfi orqali
k = 1	k = 2
	k = 3
k = 4	k = 4

Bu holda tashqi sikl **k** ning qiymati 3 bo'lguncha takrorlanishi kerak edi, lekin ichki sikl bajarilgandan so'ng **k** ning qiymati 4 ga teng bo'ldi, 4 esa 3 dan katta, jarayon to'xtaydi.

Takrorlanish tuzilmalaridan TOKI – BAJAR uchun ham ichma-ich joylashgan sikllarni tashkil etish mumkin. Zaruratga qarab turli takrorlanish tuzilmalarini ichma-ich joylashtirish ham mumkin. Masalan:

1 DAN N GACHA BAJAR
TOKI tokcha(i)=0 BAJAR
TAMOM
TAMOM

Shuni ta'kidlash kerakki, ikkita, uchta, to'rtta va hokazo sondagi takrorlanish tuzilmali ichma-ich joylashgan sikllarni tashkil etish mumkin.

8.9-mashq

Uchta takrorlanish tuzilmali algoritm yozib, jadval yordamida tahlil qiling.

Ha, aytgancha, 2 yoshga to'layotgan Bek ham bular haqida bilib olgan.

Saralash usullari

8.14-masala

Xonada N ta tokcha bo'lib, ular ustiga turli sondagi bir xil o'lchamdagi kublar ustma-ust taxlangan. Saralovchi III tokchalardagi kublarning o'rmini shunday almashtirsinki, natijada tokchalarda kublar soni kamayish tartibida joylashsin.

Avval masala shartini tushunib olaylik. Bu masalani kamayish tartibida saralash deb ham atashadi. Tokchalardagi kublar shunday joylashishi kerakki, son jihatidan qaralganda, 1-tokchaga barcha tokchalardagiga qaraganda eng ko'p kub taxlangan tokchadagi kublarni, 2-tokchaqa qolgan (o'tkazilgandan keyingi 1-tokchadagidan tashqari) barcha tokchalardagiga qaraganda eng ko'p kub taxlangan tokchadagi kublarni, 3-tokchada qolgan (o'tkazilgandan keyingi 1-2-tokchadagilardan tashqari) barcha tokchalardagiga qaraganda eng ko'p kub taxlangan tokchadagi kublarni va hokazo, o'tkazish kerak.

Jadval ko'rinishida kamayish tartibida saralashni quyidagicha tasvirlash mumkin:

8.1-jadval

Tokcha tartib raqami	1	2	3	4	5	6	7	8	9
Boshlang'ich holatda kublar soni	7	12	1	49	3	1	0	15	12
Kamayish tartibida saralanganidan keyin	49	15	12	12	7	3	1	1	0

Yechim. Biz saralashning **joylashtirish** deb ataluvchi usulidan foydalanamiz. Bu usulning ma'nosi quyidagicha: 1-tokchadan N tokchagacha qarab eng ko'p kubli tokcha aniqlanadi va undan oldingi tokchadan boshlab 1-tokchagacha har bir tokchadagi kublar bitta tokcha o'ngga suriladi, so'ng 1-tokchaga eng ko'p kub o'tkaziladi; 2-tokchadan N tokchagacha qarab eng ko'p kubli tokcha aniqlanadi va undan oldingi tokchadan boshlab 2-tokchagacha har bir tokchadagi kublar bitta tokcha o'ngga suriladi, so'ng 2-tokchaga yangi eng ko'p kub o'tkaziladi; 3-tokchadan N tokchagacha qarab eng ko'p kubli tokcha aniqlanadi va undan oldingi tokchadan boshlab 1-tokchagacha har bir tokchadagi kublar bitta tokcha o'ngga suriladi, so'ng 3-tokchaga yangi eng ko'p kub o'tkaziladi va hokazo ($N-1$)-tokchaga o'tkazilishi bilan jarayon to'xtaydi, chunki surilish va o'tkazishlar natijasida N -tokchada eng kam sondagi kublar turgan bo'ladi. Algoritmi quyidagicha:

1 DAN $N-1$ GACHA BAJAR

o'tkaz tokcha(i), Z_t

$i+1$ DAN N GACHA BAJAR

AGAR $Z_t < \text{tokcha}(j)$

U HOLDA

o'tkaz tokcha(i), Z_t

TAMOM

TAMOM {eng ko'p kubli tokcha aniqlandi}

Ek-1 DAN i GACHA BAJAR

{Saralovchi III teskaricha sanayapti}

o'tkaz tokcha(k), tokcha(k+1)

TAMOM {o'ngga bitta surilish tugadi}

o'tkaz Z_t , tokcha(i)

{eng ko'p sonli kublar i -tokchaga o'tdi}

TAMOM

Keling, **joylashtirish** usulida faqat sikllarni o'zi necha qadamligini jadval yordamida hisoblab ko'ramiz:

8.2-jadval

Tashqi sikl i	1-ichki sikl j	2-ichki sikl k	Takrorlanishlar soni
1 da	2 dan N gacha	N dan 2 gacha	$2 \cdot (N-1)$ ta
2 da	3 dan N gacha	N dan 2 gacha	$2 \cdot (N-2)$ ta
3 da	4 dan N gacha	N dan 2 gacha	$2 \cdot (N-3)$ ta
4 da	5 dan N gacha	N dan 2 gacha	$2 \cdot (N-4)$ ta
...

8.2-jadvalning davomi

N-2 da	N-1 dan N gacha	N dan N-1 gacha	$2 \cdot (N - (N-2)) = 2 \cdot 2$ ta
N-1 da	N dan N gacha	N dan N gacha	$2 \cdot (N - (N-1)) = 2 \cdot 1$ ta
Hammasi bo'lib sikllardagi qadamlar soni		$2 \cdot (1+2+3+\dots+(N-3)+(N-2)+(N-1)) = N \cdot (N-1)$ ta	

8.10-mashq

8.2-jadvaldagi qadamlar sonini hisoblashda yuqori chegara topildi. Quyidagi 8.3-jadvaldagi tokchalar uchun sikllarning qadamlari sonini hisoblang va xulosa yozing.

8.3-jadval

Tokcha tartib raqami	1	2	3	4	5
Boshlang'ich holatda kublar soni	1	3	7	12	49

8.10-mashq

8.1-jadvaldagi tokchalar uchun algoritmni tekshiring.

Biz algoritmni quyidagi jadvaldagi tokchalar uchun tekshiramiz:

8.4-jadval

Tokcha tartib raqami	1	2	3	4	5
Boshlang'ich holatda kublar soni	7	12	1	49	3

Qisqaroq bo'lishi uchun jadvalda tokcha(k) o'rniga $t(k)$ kabi yozamiz.

Tashqi sikl i	O'tkazish	Ek	Ichki sikl j	Shart tekshirish	Shart qiymati	U HOLDA o'tkazish	Ek
1	$7=t(1) \rightarrow Zt=7$	1	2	$7=Zt < t(2)=12$	ROST	$12=t(2) \rightarrow Zt=12$	2
			3	$12=Zt < t(3)=1$	YOLG'ON	—	
			4	$12=Zt < t(4)=49$	ROST	$49=t(4) \rightarrow Zt=49$	4
			5	$49=Zt < t(5)=3$	YOLG'ON	-	
			Yangi sikl k	O'tkazish			
			3	$1=t(3) \rightarrow t(4)=1$			

			2	$12 = t(2) \rightarrow t(3)=12$			
			1	$7=t(1) \rightarrow t(2)=7$			
	49= Zt $\rightarrow t(1)=49$						

Bu amallardan keyin tokchalarda kublar quyidagicha joylashadi:

Tokcha tartib raqami	1	2	3	4	5
Boshlang'ich holatda kublar soni	49	7	12	1	3

Davom etamiz:

Tash-qi sikl i	O'tkazish	Ek	Ichki sikl j	Shart tekshirish	Shart qiymati	U HOLDA o'tkazish	Ek
2	$7=t(1) \rightarrow Zt=7$	2	3	$7=Zt < t(3)=12$	ROST	$12=t(3) \rightarrow Zt=12$	3
			4	$12=Zt < t(4)=1$	YOLG'ON	—	
			5	$12=Zt < t(5)=49$	YOLG'ON	—	
			Yangi sikl k	O'tkazish			
			2	$7=t(2) \rightarrow t(3)=1$			
	$12=Zt \rightarrow t(2)=12$						

Shundan keyin tokchalarda kublar quyidagicha joylashadi:

Tokcha tartib raqami	1	2	3	4	5
Boshlang'ich holatda kublar soni	49	12	7	1	3

Keyingi qadam:

Tash-qi sikl i	O'tkazish	Ek	Ichki sikl j	Shart tekshirish	Shart qiymati	U HOLDA o'tkazish	Ek
3	$7=t(3) \rightarrow Zt=7$	3	4	$7=Zt < t(3)=1$	YOLG'ON	—	3
			5	$12=Zt < t(4)=1$	YOLG'ON	—	
			Yangi sikl k	O'tkazish			
			2	—			
	$7=Zt \rightarrow t(3)=7$						

Ma'lumki, 2 dan 3 gacha teskari sanab bo'lmaydi. Demak, yangi ichki siklda birorta ham o'tkazish bajarilmaydi, boshqacha aytganda yangi ichki sikl ishlamaydi. Jadval o'zgarmadi.

Mana oxirgi qadam:

Tashqi sikl i	O'tkazish	Ek	Ichki sikl j	Shart tekshirish	Shart qiy-mati	U HOLDA o'tkazish	Ek
4	$1=t(4) \rightarrow Zt=1$	4	5	$1=Zt < t(5)=3$	ROST	$3=t(5) \rightarrow t(4)=3$	5
			Yangi sikl k	O'tkazish			
			4	$1=t(4) \rightarrow t(5)=1$			
	$3=Zt \rightarrow t(4)=3$						

Mana endi kerakli tartib o'rnatildi:

Tokcha tartib raqami	1	2	3	4	5
Boshlang'ich holatda kublar soni	49	12	7	3	1

Bek esa quyidagicha ish tutdi: 1-tokchadan N-tokchagacha eng ko'p kub taxlangan tokchani topdi va 1-tokcha bilan almashtirdi; 2-tokchadan N-tokchagacha eng ko'p kub taxlangan tokchani topdi va 2-tokcha bilan almashtirdi; 3-tokchadan N-tokchagacha eng ko'p kub taxlangan tokchani topdi va 3-tokcha bilan almashtirdi va shu kabi almashtirishlarni (N-1)-tokchagacha davom ettirdi, chunki almashtirishlar natijasida N-tokchada eng kam sondagi kublar turgan bo'ladi.

Bek o'zi bilmagan holda **oddiy tanlov** deb ataluvchi usuldan foydalandi. Bunda u ichma-ich joylashgan sikl tashkil etdi. Tashqi va ichki sikl bir-biridan farqlanishi uchun tokcha tartib raqamida **i** va **j** harflardan foydalandi. Tokchani o'zi bilan o'zini taqqoslamalik uchun ichki siklni doimo bitta keyingi tokchadan boshladi, ya'ni tashqi sikldagi **i**-tokcha uchun ichki sikldagi sanoqni **(i+1)**-tokchadan boshlash to'g'ri bo'ladi:

1 DAN N-1 GACHA BAJAR

o'tkaz tokcha(i), Zt

i+1 DAN N GACHA BAJAR

AGAR $Zt < tokcha(j)$

U HOLDA

o'tkaz tokcha(i), Zt

TAMOM
TAMOM
o'tkaz tokcha(i), tokcha(Ek)
o'tkaz Zt, tokcha(i)

TAMOM

Joylashtirish usulidagi kabi oddiy tanlov usulida ham faqat sikllarning o'zi necha qadamligini jadval yordamida hisoblab ko'ramiz:

8.5-jadval

Tashqi sikl i	Ichki sikl j	Takrorlanishlar soni
1 da	2 dan N gacha	$1 \cdot (N-1) = N-1$ ta
2 da	3 dan N gacha	$1 \cdot (N-2) = N-2$ ta
3 da	4 dan N gacha	$1 \cdot (N-3) = N-3$ ta
4 da	5 dan N gacha	$1 \cdot (N-4) = N-4$ ta
...
N-2 da	N-1 dan N gacha	$1 \cdot (N - (N-2)) = 1 \cdot 2 = 2$ ta
N-1 da	N dan N gacha	$1 \cdot (N - (N-1)) = 1 \cdot 1 = 1$ ta
Hammasi sikllardagi bo'lib qadamlar soni		$1+2+3+\dots+(N-3)+(N-2)+(N-1) = N \cdot (N-1) : 2$ ta

Ko'rib turibsiz, oddiy tanlov usulida joylashtirish usuliga qaraganda qadamlar soni ikki marta kam ekan.

8.12-mashq

8.5-jadvaldagi qadamlar sonini hisoblashda yuqori chegara topildi. 8.3-jadvaldagi tokchalar uchun sikllarning qadamlari sonini hisoblang va xulosa yozing.

8.13-mashq

8.1-jadvaldagi tokchalar uchun algoritmni tekshiring.

Biz algoritmni 8.4-jadvaldagi tokchalar uchun tekshiramiz:

Tashqi sikl i	O'tkazish	Ek	Ichki sikl j	Shart tekshirish	Shart qiymati	U HOLDA o'tkazish	Ek
1	$7=t(1) \rightarrow Zt=7$	1	2	$7=Zt < t(2)=12$	ROST	$12=t(2) \rightarrow Zt=12$	2

			3	$12=Zt<t(3)=1$	YOLG'ON	—	2
			4	$12=Zt<t(4)=49$	ROST	$49=t(4) \rightarrow Zt=49$	4
			5	$49=Zt<t(5)=3$	YOLG'ON	—	4
	$7=t(1) \rightarrow t(4)=7$						
	$49=Zt \rightarrow t(1)=7$						

Bu amallardan keyin tokchalarda kublar quyidagicha joylashadi:

Tokcha tartib raqami	1	2	3	4	5
Boshlang'ich holatda kublar soni	49	12	1	7	3

Davom etamiz:

Tashqi sikl i	O'tkazish	Ek	Ichki sikl j	Shart tekshirish	Shart qiymati	U HOLDA o'tkazish	Ek
2	$12=t(2) \rightarrow Zt=12$	2	3	$12=Zt<t(3)=1$	YOLG'ON	—	2
			4	$12=Zt<t(4)=7$	YOLG'ON	—	2
			5	$12=Zt<t(5)=3$	YOLG'ON	—	2
	$12=t(2) \rightarrow t(2)=12$						
	$12=Zt \rightarrow t(2)=12$						

Hech narsa o'zgarmadi, davom etamiz:

Tashqi sikl i	O'tkazish	Ek	Ichki sikl j	Shart tekshirish	Shart qiymati	U HOLDA o'tkazish	Ek
3	$1=t(3) \rightarrow Zt=1$	3	4	$1=Zt<t(4)=7$	ROST	$7=t(4) \rightarrow Zt=7$	4
			5	$7=Zt<t(5)=3$	YOLG'ON	—	4
	$1=t(3) \rightarrow t(4)=1$						
	$7=Zt \rightarrow t(3)=7$						

Endi bu amallardan keyin tokchalarda kublar quyidagicha joylashadi:

Tokcha tartib raqami	1	2	3	4	5
Boshlang'ich holatda kublar soni	49	12	7	1	3

Va nihoyat oxirgi jarayon:

Tash-qi sikl i	O'tkazish	Ek	Ichki sikl j	Shart tekshirish	Shart qiymati	U HOLDA o'tkazish	Ek
4	$1=t(4) \rightarrow Zt=1$	4	5	$1=Zt < t(5)=3$	ROST	$3=t(5) \rightarrow Zt=3$	5
	$1=t(4) \rightarrow t(5)=1$						
	$3=Zt \rightarrow t(4)=3$						

Va nihoyat kerakli tartib o'rnatildi:

Tokcha tartib raqami	1	2	3	4	5
Boshlang'ich holatda kublar soni	49	12	7	3	1

Charchab ketdingiz-a! Lekin, Bek ham Siz bilan birga ishlayotganini unutmang. Cho'chimang, kichkinagina bola o'zlashtirayotgan usullar sizga ham bo'ysunadi.

To'liqlik uchun yana bir usulni ko'rib chiqamiz. Uni **oddiy almashtirish** yoki «**pufakcha**» usuli deb atashadi. Boshqa har qanday saralash usullari biz ko'rib chiqayotgan uchala usulning hosilasi bo'lar ekan.

8.14-masaladagi kamayish tartibida saralash talab qilinganda oddiy almashtirish usuli quyidagicha:

1-takrorlanishda: 1-tokcha va 2-tokchadagi kublar soni taqqoslanadi, agar 1-tokchadagi kublar soni kam bo'lsa 2-tokchaga o'tkaziladi, aks holda hech narsa qilinmaydi; 2-tokcha va 3-tokchadagi kublar soni taqqoslanadi, agar 2-tokchadagi kublar soni kam bo'lsa 3-tokchaga o'tkaziladi, aks holda hech narsa qilinmaydi va hokazo, $(N-1)$ -tokcha va N -tokchadagi kublar soni taqqoslanadi, agar $(N-1)$ -tokchadagi kublar soni kam bo'lsa N -tokchaga o'tkaziladi, aks holda hech narsa qilinmaydi. Natijada kublari soni eng kam bo'lgan tokcha N -tokchaga o'tkazilgan bo'ladi.

2-takrorlanishda: 1-tokcha va 2-tokchadagi kublar soni taqqoslanadi, agar 1-tokchadagi kublar soni kam bo'lsa 2-tokchaga o'tkaziladi, aks holda hech narsa qilinmaydi; 2-tokcha va 3-tokchadagi kublar soni taqqoslanadi, agar 2-tokchadagi kublar soni kam bo'lsa 3-tokchaga o'tkaziladi, aks holda hech narsa

qilinmaydi va hokazo, $(N-2)$ -tokcha va $(N-1)$ -tokchadagi kublar soni taqqoslanadi, agar $(N-2)$ -tokchadagi kublar soni kam bo'lsa $(N-1)$ -tokchaga o'tkaziladi, aks holda hech narsa qilinmaydi. Demak, oxirgi N -tokcha endi qaralmaydi. Natijada N -tokchadan oldingi tokchalardan kublari soni eng kam bo'lgan tokcha $(N-1)$ -tokchaga o'tkazilgan bo'ladi.

3-takrorlanishda: 1-tokcha va 2-tokchadagi kublar soni taqqoslanadi, agar 1-tokchadagi kublar soni kam bo'lsa 2-tokchaga o'tkaziladi, aks holda hech narsa qilinmaydi; 2-tokcha va 3-tokchadagi kublar soni taqqoslanadi, agar 2-tokchadagi kublar soni kam bo'lsa, 3-tokchaga o'tkaziladi, aks holda hech narsa qilinmaydi va hokazo, $(N-3)$ -tokcha va $(N-2)$ -tokchadagi kublar soni taqqoslanadi, agar $(N-3)$ -tokchadagi kublar soni kam bo'lsa $(N-2)$ -tokchaga o'tkaziladi, aks holda hech narsa qilinmaydi. Demak, oxirgi 2 ta tokcha endi qaralmaydi. Natijada $(N-1)$ -tokchadan oldingi tokchalardan kublari soni eng kam bo'lgan tokcha $(N-2)$ -tokchaga o'tkaziladi.

Shu tariqa davom ettirilsa, $(N-1)$ -takrorlanishda kerakli jadvalni hosil qilamiz. Masalada kamayish tartibida saralash so'ralgani uchun har qadamda tokchalardan kam sonlisini o'ngga surib borayapmiz. Oddiy almashtirish usulining algoritmi quyidagicha:

TAKRORLANSIN N-1 MARTA

1 DAN N-i GACHA BAJAR

AGAR tokcha(j) < tokcha(j+1)

U HOLDA

o'tkaz tokcha(j+1), Zt

o'tkaz tokcha(j), tokcha(j+1)

o'tkaz Zt, tokcha(j)

TAMOM

TAMOM

TAMOM

Algoritmdan ko'rinadiki, tashqi sikl faqat ko'riladigan tokchalar sonini kamaytirish uchun xizmat qilmoqda. Har doimgidek, «pufakcha» usulida ham faqat sikllarning o'zi necha qadamligini jadval yordamida hisoblab ko'ramiz:

8.6-jadval

Tashqi sikl i	Ichki sikl j	Takrorlanishlar soni
1 da	1 dan N-1 gacha	$1 \cdot (N-1) = N-1$ ta
2 da	1 dan N-2 gacha	$1 \cdot (N-2) = N-2$ ta

3 da	1 dan N-3 gacha	$1 \cdot (N-3) = N-3$ ta
4 da	1 dan N-4 gacha	$1 \cdot (N-4) = N-4$ ta
...
N-2 da	1 dan 2 gacha	$1 \cdot (N - (N-2)) = 1 \cdot 2 = 2$ ta
N-1 da	1 dan 1 gacha	$1 \cdot (N - (N-1)) = 1 \cdot 1 = 1$ ta
Hammasi sikllardagi bo'lib qadamlar soni		$1+2+3+\dots+(N-3)+(N-2)+(N-1) = N \cdot (N-1) : 2$ ta

8.14-mashq

8.1-, 8.4-jadvallardagi tokchalar uchun algoritmni jadval yordamida tekshiring.

7-sharh

Tokchalardagi kublar saralanganidan keyin eng ko'p kubli yoki eng kam kubli tokchani topish masalasi juda ham oson ishga aylanganini ko'rish mumkin. Haqiqatan, bu masalalarning yechimlari saralangan tokchalardagi kubli tokchalardan yoki o'ng tomondagi oxirgi tokchasi yoki chap tomondagi oxirgi tokchasi bo'ladi.

Uchala usulning sikldagi qadamlar sonini taqqoslab, «**pufakcha**» usulining boshqa usullarga nisbatan samaradorligi kam emasligini, murakkabligi esa eng kam ekanligini ko'rish mumkin.

Ko'rilgan uchala usulning samaradorligini aniqlash uchun bajariladigan amallar sonini yuqori chegarasini hisoblaymiz (* - ko'paytirish amali):

1) **Joylashtirish** usulida:

$$\begin{aligned} \text{Amallar soni} &= (N-1) + 2 \cdot ((N-1) + (N-2) + \dots + 1) + ((N-1) + (N-2) + \dots + 1) + (N-1) = \\ &= 2 \cdot (N-1) + 3 \cdot ((N-1) + (N-2) + \dots + 1) = 2 \cdot (N-1) + 3 \cdot N \cdot (N-1) : 2 = \\ &= (4 \cdot (N-1) + 3 \cdot N \cdot (N-1)) : 2 = (4 + 3 \cdot N) \cdot (N-1) : 2 \text{ ta} \end{aligned}$$

2) **Oddiy tanlov** usulida:

$$\begin{aligned} \text{Amallar soni} &= (N-1) + 2 \cdot ((N-1) + (N-2) + \dots + 1) + 2 \cdot (N-1) = \\ &= 3 \cdot (N-1) + 2 \cdot N \cdot (N-1) : 2 = (6 \cdot (N-1) + N \cdot (N-1)) : 2 = (6 + N) \cdot (N-1) : 2 \text{ ta} \end{aligned}$$

3) **Oddiy almashtirish** usulida:

$$\text{Amallar soni} = 4 \cdot ((N-1) + (N-2) + \dots + 1) = 4 \cdot N \cdot (N-1) : 2 = 2 \cdot N \cdot (N-1)$$

8.15-mashq

Yuqorida keltirilgan amallar sonining to'g'riligini o'zingiz tekshirib chiqing.

8.15-masala

Xonada N ta tokcha bo'lib, ular ustiga turli sondagi bir xil o'lchamdagi kublar ustma-ust taxlangan. Saralovchi III tokchalardagi kublarning o'rnini shunday almashtirsinki, natijada tokchalarda kublar soni o'sish tartibida joylashsin.

Yo'llanma. Taqqoslash qanday ma'noda yoziladi?

Nazorat savollari va topshiriqlar

1. Saralovchi I haqida so'zlab bering.
2. Saralovchi I ko'rsatmalari haqida so'zlab bering.
3. Saralovchi I qanday shartlar bilan ishlay oladi? Misollar yordamida izohlang.
4. Ijrochi Saralovchi I uchun INKOR holatlarga misollar keltiring.
5. Saralovchi II haqida so'zlab bering.
6. Saralovchi II ko'rsatmalari haqida so'zlab bering.
7. Saralovchi II qanday shartlar bilan ishlay oladi? Misollar yordamida izohlang.
8. Ijrochi Saralovchi II uchun INKOR holatlarga misollar keltiring.
9. Saralovchi III haqida so'zlab bering.
10. Saralovchi III ko'rsatmalari haqida so'zlab bering.
11. Saralovchi III qanday shartlar bilan ishlay oladi? Misollar yordamida izohlang.
12. Ijrochi Saralovchi III uchun INKOR holatlarga misollar keltiring.
13. Saralsh deganda nimani tushunasiz?
14. Qanday saralash usullari bor ekan?
15. Joylashtirish usulini o'zingiz hosil qilgan tokchalar orqali tushuntiring.
16. Oddiy tanlov usulini o'zingiz hosil qilgan tokchalar orqali tushuntiring.
17. Oddiy almashtirish usulini o'zingiz hosil qilgan tokchalar orqali tushuntiring.
18. Saralash usullarining samaradorligi va murakkabligi haqida ma'lumot bering.
19. Bobdagi barcha vazifalarni bajaring.

Qo'shimcha masalalar

S-I-8.1. A , B , D tokchalarda buyum turibdi. E tokcha bo'sh. Tokchalardagi buyumlarni siklik o'ngga suring, ya'ni buyumlarni A dan B ga, B dan D ga, D dan A ga o'tkazilsin.

S-I-8.2. A tokchada 1 ta, B tokchada 2 ta, D tokchada 3 ta kub bor. E tokcha bo'sh. Tokchalardagi kublarni shunday o'tkazish kerakki, natijada g'oliblar shohsupasi hosil bo'lsin.

S-I-8.3. A tokchada 1 ta, B tokchada 2 ta, D tokchada 3 ta kub bor. E tokcha bo'sh. Tokchalardagi kublarni bir-biriga shunday o'tkazish kerakki, natijada pastga olib tushadigan zinapoya hosil bo'lsin.

S-II-8.4. $2N$ ta tokchada buyumlar bor. $(2N+1)$ -tokcha bo'sh. Toq tartib raqamli tokchalardagi buyumlar bilan juft tartib raqamli tokchalardagi buyumlarning o'рни almashtirilsin.

S-II-8.5. Xonada 99 ta tokcha bo'lib, ularning ustiga turli og'irlikdagi buyumlar qo'yilgan. Saralovchi II eng og'ir buyumni aniqlashi kerak.

S-II-8.6. Xonada 99 ta tokcha bo'lib, ularning ustiga turli og'irlikdagi buyumlar qo'yilgan. Saralovchi II eng yengil buyumni aniqlashi kerak.

S-II-8.7. Xonada N ta tokcha bo'lib, ulardan M tasi ustiga turli og'irlikdagi buyumlar qo'yilgan, qolganlari bo'sh. Saralovchi II eng og'ir buyumni aniqlashi kerak.

S-II-8.8. Xonada N ta tokcha bo'lib, ulardan M tasi ustiga turli og'irlikdagi buyumlar qo'yilgan, qolganlari bo'sh. Saralovchi II eng yengil buyumni aniqlashi kerak.

S-II-8.9. Xonada N ta tokcha bo'lib, ular ustiga turli og'irlikdagi buyumlar qo'yilgan, ba'zilari bo'sh. Saralovchi II bo'sh tokchani aniqlashi kerak.

S-III-8.10. Xonada N ta tokcha bo'lib, ular ustiga turli og'irlikdagi buyumlar qo'yilgan, ba'zilari bo'sh. Saralovchi III bo'sh tokchani aniqlashi kerak.

S-III-8.11. Xonada 99 ta tokcha bo'lib, ularning ustiga turli og'irlikdagi buyumlar qo'yilgan. Saralovchi III eng og'ir buyum turgan tokchani tartib raqamini aniqlashi kerak.

S-III-8.12. Xonada 99 ta tokcha bo'lib, ularning ustiga turli og'irlikdagi buyumlar qo'yilgan. Saralovchi III eng yengil buyum turgan tokchani tartib raqamini aniqlashi kerak.

S-III-8.13. Xonada N ta tokcha bo'lib, ulardan M tasi ustiga turli og'irlikdagi buyumlar qo'yilgan, qolganlari bo'sh. Saralovchi III eng og'ir buyum turgan tokchaning tartib raqamini aniqlashi kerak.

S-III-8.14. Xonada N ta tokcha bo'lib, ulardan M tasi ustiga turli og'irlikdagi buyumlar qo'yilgan, qolganlari bo'sh. Saralovchi III eng yengil buyum turgan tokchaning tartib raqamini aniqlashi kerak.

S-III-8.15. Xonada N ta tokcha bor. Ular ustiga turli og'irlikdagi buyumlar qo'yilgan. Saralovchi III K -tokchadagi kublar bilan va M -tokchadagi kublarning o'rnini almashtirsin.

S-III-8.16. Xonada N ta tokcha bor. Ular ustiga turli og'irlikdagi buyumlar qo'yilgan. Saralovchi III eng yengil buyum bilan eng og'ir buyumning o'rnini almashtirsin.

Yo'llanma. Avval saralash, keyin almashtirish kerak.

S-III-8.17. Xonada N ta tokcha bor. Ular ustiga turli og'irlikdagi buyumlar qo'yilgan. Saralovchi III eng yengil buyumli tokchalar sonini aniqlasin.

Yo'llanma. Masalan, avval 0 dan farqli eng yengili topiladi, keyin eng yengilidan katta og'irlikdagi buyumli tokchalar bo'shatiladi, tokchalarni kamayish tartibida saralanadi, so'ng 1-bo'sh tokchadan oldingi tokcha tartib raqami aniqlanadi.

S-III-8.18. Xonada N ta tokcha bor. Ular ustiga turli og'irlikdagi buyumlar qo'yilgan. Saralovchi III eng og'ir buyumli tokchalar sonini aniqlasin.

S-III-8.19. Xonada N ta tokcha bor. Ular ustiga turli og'irlikdagi buyumlar qo'yilgan. Saralovchi III bo'sh bo'lmagan tokchalar sonini aniqlasin.

Yangi imkoniyatlar

Hozirgacha ko'p imkoniyatli hisoblangan **bir qo'lli**, **Zt** nomli zaxira tokchali, natural son o'tkazish mumkin bo'lgan **Ek** nomli ekranchasi bor g'ildirakli Saralovchi III quyidagi imkoniyatlarga ega edi:

- **Ijrochi muhiti** – devorlariga soni quyi INF va yuqori SUP chegaraga ega $INF=1$ dan SUP gacha tartib raqami berilgan: tokcha(1), tokcha(2), ..., tokcha(7), ..., tokcha(21), ..., tokcha(1963), ...

ba'zilarini yopib qo'yish imkoni bo'lgan tokchalar o'rnatilgan doira shaklidagi xona;

- **Ijrochining ko'rsatmalar tizimi** – quyidagi ko'rsatmalardan iborat:

o'tkaz tokcha(N), tokcha(M)

o'tkaz tokcha(N), Zt

o'tkaz Zt, tokcha(M);

- **Bajara oladigan amallari** – buyumni bir tokchadan boshqa tokchaga o'tkazishda o'tkazilayotgan tokchani bo'shatib o'tkazish; Zt tokchaga ixtiyoriy tokchanning nusxasini olish ma'nosida o'tkazish; ixtiyoriy tokchada Zt tokchanning nusxasini o'tkazish; Ek nomli ekranchada Zt tokchadagi buyumning tartib raqamini aks ettirish va Ek ga biror natural son o'tkazilsa Zt tokchada shu songa mos tartib raqamli tokchanning nusxasini o'tkazish; arifmetik amallarni bajara olish; sonlar va tokchalardagi buyumlarni miqdoriy kattaliklari ustida tekshirilayotgan quyidagi oddiy

= (teng), <>(teng emas), <(kichik), <=(katta emas), >(katta), >= (kichik emas)

shartlarni va mantiqiy amallar asosida hosil qilingan birikkan shartlarni tekshirib mantiqiy xulosa chiqara olish, barcha takrorlash tuzilmalarini bajara olish.

Unga algoritmlarni yozadigan inson Bek nomli bola bo'lib, uni qilgan ishlarini munosib baholab, endi **dasturchi** deb atasak

ham bo'ladi. Saralovchi III imkoniyatlarini oshirish, ya'ni yangi muhit barpo etish yoki dasturiga o'zgartirish kiritish yoki tuzilishiga yangi qismlar qo'shish Bekning ota-onasi xohishiga bog'liq edi. Qarangki, o'g'illari 2 yoshga to'lganda quyidagi imkoniyatlar bilan boyitib **Saralovchi M** nomli yangi ijrochi hosil qilishdi:

- **Ijrochi muhiti** – devorlariga soni quyi va yuqori chegaraga ega tartib raqamlari berilgan ko'p qavatli (endi qulaylik uchun **tokcha(k)** o'rniga **t(k)** deb yozamiz):

$t1(1), t1(2), \dots, t1(7), \dots, t1(21), \dots, t1(n)$ (1-qavat)

$t2(1), t2(2), \dots, t2(4), \dots, t2(20), \dots, t2(m)$ (2-qavat)

... ..

$tp(1), tp(2), \dots, tp(5), \dots, tp(10), \dots, tp(k)$ (p-qavat)

ba'zilarini yopib qo'yish imkoni bo'lgan tokchalar o'rnatilgan doira shaklidagi xona;

- **Ijrochining ko'rsatmalar tizimi** – quyidagi ko'rsatmalardan iborat:

o'tkaz $tx(N), ty(M)$

o'tkaz $tx(N), Zt$

o'tkaz $Zt, tx(M)$

o'tkaz $tx(N)+A, tz(K)$

o'tkaz $tx(N)-A, tz(K)$

o'tkaz $tx(N) \cdot A, tz(K)$

o'tkaz $tx(N)/A, tz(K)$

o'tkaz $A+ty(M), tz(K)$

o'tkaz $A-ty(M), tz(K)$

o'tkaz $A \cdot ty(M), tz(K)$

o'tkaz $A/ty(M), tz(K)$

bo'shat A

bunda x, y ixtiyoriy qavat tartib raqamlari, \cdot – ko'paytirish va $/$ – bo'lish amallari, A – qandaydir tokcha yoki son;

- **Bajara oladigan amallari** – Saralovchi III ning amallariga qo'shimcha tokchadagi buyumlar ustida tokchani bo'shatish, ya'ni **bo'shat A** , miqdoriy qo'shish (oshirish), ayirish (kamaytirish), ko'paytirish (marta oshirish) va bo'lish (marta kamaytirish) amallari:

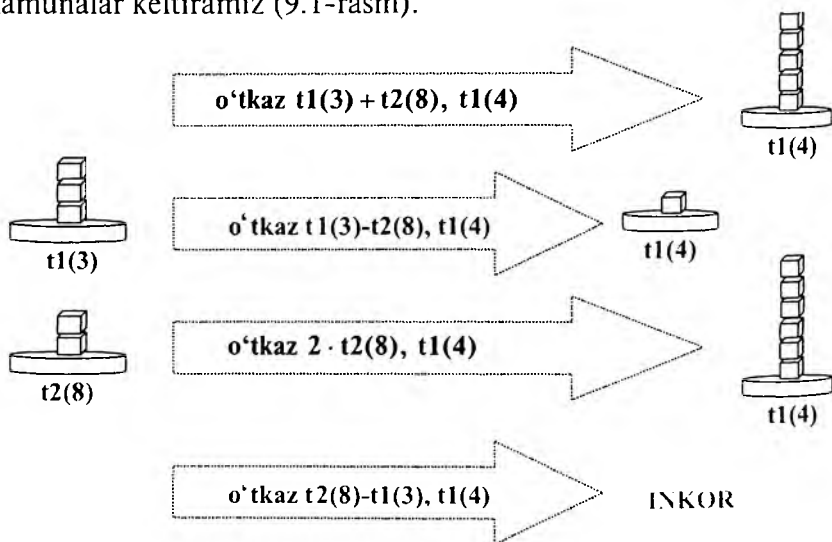
$tx(n)+ty(m), tx(n)-ty(m), tx(n) \cdot ty(m), tx(n)/ty(m), tx(n)+A,$

$tx(n)-A, tx(n) \cdot A, tx(n)/A, A+tx(n), A-tx(n), A \cdot tx(n), A/tx(n),$

$tx(n)+Zt, \dots;$

- Yuqoridagi amallarni bajara olmasa **INKOR** holat yuzaga keladi, masalan, **o'tkaz $t2(5) - t1(4), t3(2)$** ko'rsatmasi **$t2(5) < t1(4)$** bo'ladi, chunki manfiy sondagi buyum yo'q.

Ijrochi Saralovchi M ning bajaradigan amallarini tushunib olish lozim bo'ladi. Agar, masalan, **o'tkaz $t_2(5)+t_1(4)$, $t_3(2)$** ko'rsatmasida 2-qavatdagi 5-tokchadagi buyum nusxasi miqdoriga 1-qavatdagi 4-tokchadagi buyum nusxasi miqdori qo'shilib natijaga mos miqdordagi buyum 3-qavatdagi 2-tokchaga o'tkazilsa, **o'tkaz $21/t_1(4)$, $t_3(2)$** ko'rsatmasida 21 soni 1-qavatdagi 4-tokchadagi buyum miqdoriga bo'linib, natijaning miqdoricha buyum nusxasi 3-qavatdagi 2-tokchaga o'tkaziladi. Albatta, bu hollarda **$t_3(2)$** tokchadagi buyum avval tashlab yuborilib, keyin natija miqdoricha buyum nusxasi o'tkaziladi. Agar **$t_3(2)$** tokchadagi buyumni ham shu tokchada saqlab qolish kerak bo'lsa, u holda ko'rsatmani **o'tkaz $t_2(5)+t_1(4)+t_3(2)$, $t_3(2)$** yoki **o'tkaz $21/t_1(4)+t_3(2)$, $t_3(2)$** kabi yozish kerak bo'ladi. Quyida bu amallarga mos rasmiy namunalari keltiramiz (9.1-rasm).



9.1-rasm.

Qulaylik uchun bundan keyin **x -qavatdagi tokchalar guruhini $t_x(*)$** kabi belgilaymiz. Qavatdagi tokchalar soni alohida aytib o'tiladi.

Bu imkoniyatlari bilan Saralovchi M qanday masalalarni hal etishga qodirligini ko'ramiz.

Sodda masalalar

9.1-masala

Saralovchi M 5 ta tokchali $t_1(*)$ ni barcha buyumlarini S tokchaga yig'sin.

Yechim. Bek avval *S* tokchani bo'shatib oldi, chunki xato natijaga olib kelmasligi uchun unda birorta buyum bo'lmisligi kerak. Keyin yuqoridagi ma'lumotlarga amal qilib har bir tokchadagi buyumni birma-bir *S* tokchaga o'tkazib chiqdi:

bo'shat *S*

o'tkaz $S+t1(1)$, *S*

o'tkaz $S+t1(2)$, *S*

o'tkaz $S+t1(3)$, *S*

o'tkaz $S+t1(4)$, *S*

o'tkaz $S+t1(5)$, *S*

Juda oson, shunday emasmi? Har bir qadamni izohlab chiqaylik.

Ko'rsatmalar	Saralovchi bajargan amallar
bo'shat <i>S</i>	<i>S</i> tokcha bo'shatildi, ya'ni buyum miqdori 0 ga teng
o'tkaz $S+t1(1)$, <i>S</i>	<i>S</i> tokchadagi buyum nusxasi miqdori 0 ga 1-qavatning 1-tokchasidagi buyum nusxasi miqdori t1(1) qo'shildi va natijaga mos t1(1) miqdordagi buyum <i>S</i> tokchaga qoidaga binoan o'tkazildi, ya'ni o'tkazishda avval <i>S</i> tokchada turgan buyum tashlab yuborildi va yangi miqdordagi buyum qo'yildi
o'tkaz $S+t1(2)$, <i>S</i>	<i>S</i> tokchadagi buyum nusxasi miqdori t1(1) ga 1-qavatning 2-tokchasidagi buyum nusxasi miqdori t1(2) qo'shildi va natijaga mos t1(1)+t1(2) miqdordagi buyum <i>S</i> tokchaga qoidaga binoan o'tkazildi, ya'ni o'tkazishda avval <i>S</i> tokchada turgan buyum tashlab yuborildi va yangi miqdordagi buyum qo'yildi
o'tkaz $S+t1(3)$, <i>S</i>	<i>S</i> tokchadagi buyum nusxasi miqdori t1(1)+t1(2) ga 1-qavatning 3-tokchasidagi buyum nusxasi miqdori t1(3) qo'shildi va natijaga mos t1(1)+t1(2)+t1(3) miqdordagi buyum <i>S</i> tokchaga qoidaga binoan o'tkazildi, ya'ni o'tkazishda avval <i>S</i> tokchada turgan buyum tashlab yuborildi va yangi miqdordagi buyum qo'yildi
o'tkaz $S+t1(4)$, <i>S</i>	<i>S</i> tokchadagi buyum nusxasi miqdori t1(1)+t1(2)+t1(3) ga 1-qavatning 4-tokchasidagi buyum nusxasi miqdori t1(4) qo'shildi va natijaga mos t1(1)+t1(2)+t1(3)+t1(4) miqdordagi buyum <i>S</i> tokchaga qoidaga binoan o'tkazildi, ya'ni o'tkazishda avval <i>S</i> tokchada turgan buyum tashlab yuborildi va yangi miqdordagi buyum qo'yildi
o'tkaz $S+t1(5)$, <i>S</i>	<i>S</i> tokchadagi buyum nusxasi miqdori t1(1)+t1(2)+t1(3)+t1(4) ga 1-qavatning 5-tokchasidagi buyum nusxasi miqdori t1(5) qo'shildi va natijaga mos t1(1)+t1(2)+t1(3)+t1(4)+t1(5) miqdordagi buyum <i>S</i> tokchaga qoidaga binoan o'tkazildi, ya'ni o'tkazishda avval <i>S</i> tokchada turgan buyum tashlab yuborildi va yangi miqdordagi buyum qo'yildi

9.1-mashq

Algoritmi takrorlash tuzilmasi yordamida yozing.

9.2-mashq

Saralovchi M 5 ta tokchali $t1(*)$ ni barcha buyumlari miqdori-ning ko'paytmasini S tokchaga o'tkazsin.

Yo'llanma. Yuqoridagi algoritmda bo'shatish o'rniga ko'paytirishda ko'paytma qiymatiga ta'sir etmaydigan miqdor (nechaga teng bo'ladi?) S ga o'tkaziladi va qo'shishni ko'paytirishga almash-tiladi.

9.2-masala

Saralovchi M N ta tokchali $t1(*)$ ni barcha buyumlarini S tokchaga yig'sin.

Yechim. Endi tokchalar soni (ko'p) N ta bo'lgani uchun takrorlash tuzilmasiz masalani hal etib bo'lmaydi. Buni tushungan Bek Saralovchi M uchun masalaning 2 xil ko'rinishdagi algoritmini yozdi:

bo'shat S
TAKRORLANSIN N MARTA
o'tkaz S+t1(i), S
TAMOM

va

bo'shat S
1 DAN N GACHA BAJAR
o'tkaz S+t1(i), S
TAMOM

Buncha qisqa, to'g'ri natija berarmikan?

9.3-mashq

Algoritm to'g'ri ishlashini misol yordamida tekshirib ko'ring.

9.4-mashq

Saralovchi M N ta tokchali $t1(*)$ ni barcha buyumlari miq- dorining ko'paytmasini S tokchaga o'tkazsin.

9.3-masala

Saralovchi M N ta tokchali $t1(*)$ ni M -tokchasidan K -tokcha- sigacha bo'lgan barcha buyumlarini S tokchaga yig'sin.

Yechim. Avvalgi masala yechimini tuzayotganda Bek shunday masala berilishini sezgan ekan. Nega desangiz, bu masalaning

yechimini hosil qilish uchun ikkinchi ko‘rinishdagi algoritmni o‘ziga o‘zgartirish kifoya:

bo‘shat S

M DAN K GACHA BAJAR

{tartib raqamlari M dan K gacha}

o‘tkaz S+t1(i), S

TAMOM

Masalaning yechimini boshqa takrorlanish tuzilmasi orqali ham tashkil etish mumkin:

bo‘shat S

TAKRORLANSIN N MARTA

AGAR $M \leq i$ VA $i \leq K$

{tartib raqamlari M dan K gacha}

U HOLDA

o‘tkaz S+t1(i), S

TAMOM

TAMOM

Mana dasturchilarning mantiqi va iqtidori! Ularga bir masala bersangiz shu masala atrofidagi masalalarning o‘zlari tuzib olib yechaverishadi yoki keyingi masala qanday bo‘lishini oldindan sezishadi!

9.5-mashq

Saralovchi M N ta tokchali $t1(*)$ ni M -tokchasidan K -tokchasigacha bo‘lgan barcha buyumlari miqdorini ko‘paytmasini S tokchaga o‘tkazsin.

9.4-masala

Saralovchi M N ta tokchali $t1(*)$ ni buyumlari miqdori M va K orasida bo‘lgan tokchalarining barcha buyumlarini S tokchaga yig‘sin.

Yechim. Bu masala ham Bek uchun yengil bo‘lsa kerak. Algoritm takrorlash va shart tekshirish yordamida quyidagicha tuzilgan:

bo‘shat S

TAKRORLANSIN N MARTA

AGAR $M \leq t1(i)$ VA $t1(i) \leq K$

{buyum miqdori oraliqda bo‘lishi}

U HOLDA

o‘tkaz S+t1(i), S

TAMOM

TAMOM

9.6-mashq

Saralovchi $M N$ ta tokchali $t1(*)$ ni buyumlari miqdori M va K orasida bo'lgan tokchalarini barcha buyumlari miqdorining ko'paytmasini S tokchaga o'tkazsin.

9.5-masala

Saralovchi $M N$ ta tokchali $t1(*)$ ni buyumlari miqdori M va K orasida bo'lgan tokchalarining barcha buyumlarini $S1$ tokchaga, qolganlarini $S2$ tokchaga yig'sin.

Yechim. Ha, mana, Bek uchun iqtidoriga yarasha qiziqarliroq masala. Takrorlash va tarmoqlanish aralashgan algoritm shunday:

bo'shat $S1$

bo'shat $S2$

TAKRORLANSIN N MARTA

AGAR $M \leq t1(i)$ VA $t1(i) \leq K$

{buyum miqdori oraliqda bo'lishi}

U HOLDA

o'tkaz $S1+t1(i)$, $S1$

{buyum miqdori oraliqda bo'lganlar}

AKS HOLDA

o'tkaz $S2+t1(i)$, $S2$

{buyum miqdori oraliqda bo'lmaganlar}

TAMOM

TAMOM

9.7-mashq

Saralovchi $M N$ ta tokchali $t1(*)$ ni buyumlari miqdori M dan kichik bo'lgan tokchalarining barcha buyumlarini $S1$ tokchaga, qolganlarini $S2$ tokchaga yig'sin.

9.8-mashq

Saralovchi $M N$ ta tokchali $t1(*)$ ning buyumlari miqdori M dan kichik bo'lgan tokchalarining barcha buyumlarini $S1$ tokchaga, N ta tokchali $t1(*)$ ni buyumlari miqdori M va K orasida bo'lgan tokchalarining barcha buyumlarini $S2$ tokchaga va qolganlarini $S3$ tokchaga yig'sin.

9.9-mashq

Saralovchi $M N$ ta tokchali $t1(*)$ ning tartib raqamlari M dan kichik bo'lgan tokchalarining barcha buyumlarini $S1$ tokchaga, N ta tokchali $t1(*)$ ni tartib raqamlari M va K orasida bo'lgan tokchalarining barcha buyumlarini $S2$ tokchaga va qolganlarini $S3$ tokchaga yig'sin.

9.6-masala

Saralovchi M juft N ta tokchali $t1(*)$ ning tartib raqamlari juft bo'lgan tokchalarining barcha buyumlarini S tokchaga yig'sin.

Yechim. Bek ham charchamaydigan bola ekanda, masalalarni yechmay turganda ozgina dam olardik. Bu masalani yecha olmasa kerak, chunki Saralovchi M juft degan tushunchani ham, juft degan shartini ham tushunmaydi va tekshira olmaydi. Uni ixtiyorida hammasi bo'lib to'rt arifmetik amal va bob boshida bayon etilgan shart tekshirishlar bor. Juft degan shartni tekshirish uchun, odatda, sonni 2 ga bo'lganda qoldiq qolmasligi orqali ($\text{son mod } 2 = 0$) yoki sonni 2 ga bo'lganda butun chiqish talabi ($\lfloor \text{son}/2 \rfloor = \text{son}/2$) orqali tekshiriladi. Bek mana qanday algoritm tuzibdi:

bo'shat S

TAKRORLANSIN N/2 MARTA

o'tkaz $S+t1(2\cdot i)$, S

TAMOM

Qoyil, buni ham yo'lini topdi. Algoritm mazmuniga e'tibor bering: masala shartida N juft natural son va shuning uchun ham uni 2 ga bo'lganda natural son hosil bo'ladi, ya'ni sanashda **INKOR** yuzaga kelmaydi. Takrorlanishda sanoq $i = 1$ bo'lsa: $2 \cdot i = 2$ – birinchi juft son; sanoq $i = 2$ bo'lsa: $2 \cdot i = 4$ – ikkinchi juft son; sanoq $i = 3$ bo'lsa: $2 \cdot i = 6$ – uchunchi juft son; ...; sanoq $i = N/2$ bo'lsa: $2 \cdot i = N$ – oxirgi juft son hosil bo'ladi.

9.10-mashq

Saralovchi M toq N ta tokchali $t1(*)$ ning tartib raqamlari toq bo'lgan tokchalarini barcha buyumlarini S tokchaga yig'sin.

9.7-masala

Saralovchi M N ta tokchali $t1(*)$ ning tartib raqamlari juft bo'lgan tokchalarining barcha buyumlarini S tokchaga yig'sin.

Yechim. Bu masalada N ning toqligi ham juftligi ham ma'lum emas. Endi bu vaziyatdan qanday chiqib ketish mumkin? Qani Bek tuzgan algoritmni qarab chiqaylik-chi:

bo'shat S

TAKRORLANSIN N MARTA

AGAR $2\cdot i \leq N$

{juft sonni yuqori chegaradan oshib ketmaslik sharti}

U HOLDA

o'tkaz $S+t1(2\cdot i)$, S

TAMOM

TAMOM

Nima ham derdik, algoritm masala shartiga to'liq javob beradi. Agar N juft bo'lsa $N/2$ natural ekanligi, N toq bo'lsa $(N-1)/2$ natural ekanligini hisobga olsak, quyidagi jadval algoritmni tushunishga oydinlik kiritadi:

N juft, Sanoq i	Juft son: 2·i	N toq, Sanoq i	Juft son: 2·i
1	2	1	2
2	4	2	4
3	6	3	6
...
$N/2$	N	$(N-1)/2$	$N-1$
Shu qadamda jarayon to'xtaydi, chunki:			
$N/2+1$	$N+2 > N$	$(N+1)/2$	$N+1 > N$

9.11-mashq

Saralovchi M N ta tokchali $t1(*)$ ning tartib raqamlari juft bo'lgan tokchalarining barcha buyumlarini $S1$ tokchaga, tartib raqamlari toqlarini $S2$ tokchaga yig'sin.

9.12-mashq

Saralovchi M N ta tokchali $t1(*)$ ning tartib raqamlari juft bo'lgan tokchalarini barcha buyumlari miqdorining ko'paytmasini $S1$ tokchaga, tartib raqamlari toq tokchalari barcha buyumlari miqdorining ko'paytmasini $S2$ tokchaga o'tkazsin.

9.13-mashq

Saralovchi M N ta tokchali $t1(*)$ ni tartib raqamlari 5 ga karrali bo'lgan tokchalarini barcha buyumlari miqdorining ko'paytmasini $S1$ tokchaga, tartib raqamlari 2 ga karrali bo'lgan tokchalari barcha buyumlari miqdorining ko'paytmasini $S2$ tokchaga o'tkazsin.

9.8-masala

Saralovchi M N ta tokchali $t1(*)$ ni buyumlarining miqdori 20 dan kichik bo'lgan tokchalarining sonini aniqlasin.

Yo'llanma. $t1(*)$ ni o'sish tartibida saralash kerak.

9.9-masala

Saralovchi M N ta tokchali $t1(*)$ ni buyumlari miqdori M va K orasida bo'lgan tokchalarining buyumini 1 ta buyumga almashtirsin.

9.14-mashq

Saralovchi M N ta tokchali $t1(*)$ ni buyumlari miqdori M va K orasida bo'lgan tokchalarini bo'shatsin.

9.15-mashq

Saralovchi M N ta tokchali $t1(*)$ ning bo'sh tokchalari sonini aniqlasin.

9.16-mashq

Saralovchi M N ta tokchali $t1(*)$ ning kublari soni 5 ga karrali bo'lgan tokchalarini bo'shatsin.

9.17-mashq

Saralovchi M N ta tokchali $t1(*)$ ni buyumlari miqdori teng bo'lgan tokchalardan ketma-ket kelganlarini bo'shatsin.

Yo'llanma. Avval tokchalar saralanadi, bu holda buyumlari miqdori teng bo'lgan tokchalar ketma-ket joylashadi. So'ng bo'sh S tokchani kerakli tokchalarga o'tkaziladi.

Qiyinroq masalalar

Umumiylikni yo'qotmagan holda hamda qulaylik uchun bu bo'limda tokchalardagi buyumlar o'rniga ularni miqdori yoki sonlar turibdi, deb olishimiz mumkin. Ya'ni, tokchada 5 ta kub o'rniga 5 soni, 10 kg yuk o'rniga 10 soni turibdi, deyishimiz mumkin.

9.10-masala

N ta tokchali $t1(*)$ va $t2(*)$ qavatlar berilgan. S tokchaga $t1(i) \cdot t2(j)$ juftliklarni shunday tashkil etib yig'ish kerakki, S ning qiymati boshqa juftliklarga nisbatan eng katta bo'lsin, ya'ni $t1(*)$ va $t2(*)$ qavatlar tokchalari $t(i) \cdot t(j)$ ko'paytmalarining yig'indisi eng katta bo'lsin.

Yechim. Masalani avval misol ko'rish orqali tushunib olamiz:

Tartib raqamlar	1	2	3	4
$t1(*)$	3	0	2	7
$t2(*)$	5	9	3	3
$S = t1(1) \cdot t2(1) + t1(2) \cdot t2(2) + t1(3) \cdot t2(3) + t1(4) \cdot t2(4) = 15 + 0 + 6 + 21 = 42$				

$S = t1(1) \cdot t2(2) + t1(2) \cdot t2(1) + t1(3) \cdot t2(4) + t1(4) \cdot t2(3) = 27 + 0 + 6 + 21 = 48$
$S = t1(1) \cdot t2(3) + t1(2) \cdot t2(4) + t1(3) \cdot t2(2) + t1(4) \cdot t2(1) = 9 + 0 + 18 + 35 = 62$
$S = t1(1) \cdot t2(4) + t1(2) \cdot t2(2) + t1(3) \cdot t2(1) + t1(4) \cdot t2(3) = 9 + 0 + 10 + 21 = 40$
$S = t1(1) \cdot t2(1) + t1(2) \cdot t2(3) + t1(3) \cdot t2(4) + t1(4) \cdot t2(2) = 15 + 0 + 6 + 63 = 84$
... ..

Yana davom ettirish mumkin. Lekin bu usulda hamma holni ko'rib chiqish ancha murakkab bo'lar ekan. Shuning uchun matematikaga murojaat etamiz:

Agar $A \leq B$ va $C \leq D$ bo'lsa, u holda

$$A \cdot C + B \cdot D \geq A \cdot D + B \cdot C$$

bo'ladi.

Haqiqatan, $A - B \leq 0$ va $C - D \leq 0$ bo'lgani uchun:

$$\begin{aligned} (A - B) \cdot (C - D) \geq 0 &\Leftrightarrow A \cdot (C - D) - B \cdot (C - D) \geq 0 \Leftrightarrow \\ \Leftrightarrow A \cdot C + B \cdot D - A \cdot D - B \cdot C \geq 0 &\Leftrightarrow A \cdot C + B \cdot D \geq A \cdot D + B \cdot C \end{aligned}$$

Demak, juftliklar ko'paytmasining yig'indisi eng katta qiymatli bo'lishi uchun juftliklarni bir xil tartibda (yoki ikkalasini o'sish tartibida, yoki ikkalasini kamayish tartibida) saralash lozim.

Bu xulosaga asosan tuzilgan algoritm quyidagicha bo'ladi:

TAKRORLANSIN N-1 MARTA

1 DAN N-i GACHA BAJAR

AGAR $t1(j) < t1(j+1)$

{ $t1(*)$ }-qavatdagi tokchalarni saralash}

U HOLDA

o'tkaz $t1(j+1)$, Z_t

o'tkaz $t1(j)$, $t1(j+1)$

o'tkaz Z_t , $t1(j)$

TAMOM

AGAR $t2(j) < t2(j+1)$

{ $t2(*)$ }-qavatdagi tokchalarni saralash}

U HOLDA

o'tkaz $t2(j+1)$, Z_t

o'tkaz $t2(j)$, $t2(j+1)$

o'tkaz Z_t , $t2(j)$

TAMOM
TAMOM
TAMOM
bo'shat S
TAKRORLANSIN N MARTA {yig'indini hosil qilish}
o'tkaz $S+t1(i) \cdot t2(i)$, S
TAMOM

9.17-mashq

N ta tokchali $t1(*)$ va $t2(*)$ qavatlar berilgan. Saralovchi M $t1(i) \cdot t2(j)$ juftliklarni shunday tashkil etib S tokchaga yig'ishi kerakki, S ning qiymati boshqa juftliklarga nisbatan eng kichik bo'lsin, ya'ni $t1(*)$ va $t2(*)$ qavatlar tokchalari $t(i) \cdot t(j)$ ko'paytmalarining yig'indisi eng kichik bo'lsin.

9.11-masala

Son polindrom deyiladi, agar uni chapdan ham, o'ngdan ham bir xil o'qilsa. N ta tokchali $t1(*)$ ning har bir tokchasida bittadan raqam bor. Saralovchi M $t1(*)$ qavat polindrom bo'lsa, Zt ga 1 ni, $t1(*)$ qavat polindrom bo'lmasa 0 ni o'tkazsin.

Yechim. Ta'rifga ko'ra, 12621 – polindrom, 1262 – polindrom emas. Ma'lumki, raqam faqat bir xonali bo'ladi, ya'ni masalan, o'nlik sanoq sistemasida 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 lar raqam. Bek Saralovchi M uchun shunday algoritm tuzdi: avval $t1(*)$ qavatni teskari tartibda $t2(*)$ qavatga nusxaladi, ya'ni $t1(*)$ qavatdagi **i-tokcha** $t2(*)$ qavatdagi **(N-i+1)-tokchaga** nusxalandi. Keyin $t1(*)$ va $t2(*)$ qavatlarning bir xil tartib raqamli tokchalaridagi raqamlar tengligini tekshirdi va algoritm yakunida olingan natijaga qarab xulosa chiqardi.

9.18-mashq

Masala algoritmini mustaqil yozing.

9.12-masala

Saralovchi M N ta tokchali $t1(*)$ va $t2(*)$ ni barcha sonlarini S tokchaga yig'sin.

9.13-masala

Saralovchi M N ta tokchali $t1(*)$ ni tokchalaridagi barcha sonlarning o'rta arifmetigini S tokchaga yig'sin.

9.19-mashq

Informatika fanidan har kuni olgan ballaringizni tokchlarga tartib bilan joylashtirib chiqib, yakuniy natija balingizni aniqlovchi algoritim tuzing.

9.14-masala

Saralovchi M berilgan N uchun 2 ning darajalarini $t1(*)$ ni tokchalariga joylashtirib, keyin ularni S tokchaga yig' sin.

Yechim. Masala shartiga ko'ra Bek algoritmidagi $t1(1)$ tokchaga 2 ning 1-darajasini joylashtirdi; $t1(2)$ tokchaga 2 ning 2-darajasini joylashtirdi va hokazo:

1 DAN N GACHA BAJAR

o'tkaz 2^i , $t1(i)$

TAMOM

yoki boshqacha takrorlanish tuzilmasi orqali yozsak:

TAKRORLANSIN N MARTA

o'tkaz 2^i , $t1(i)$

TAMOM

Bunday masalalarda keyingi qadamni bitta oldingi qadamga bog'liq ravishda hisoblash ham mumkin: $t1(1)$ tokchaga 2 ning 1-darajasini joylashtiramiz; $t1(2)$ tokchaga $t1(1)$ ni 2 ga ko'paytirib joylaymiz, ya'ni 2 ni 2 ga ko'paytirib $t1(2)$ tokchaga joylashtiramiz; $t1(3)$ tokchaga $t1(2)$ ni 2 ga ko'paytirib joylaymiz, ya'ni $2 \cdot 2$ ni 2 ga ko'paytirib $t1(3)$ tokchaga joylashtiramiz va hokazo:

o'tkaz 2, $t1(1)$

2 DAN N GACHA BAJAR

o'tkaz $2 \cdot t1(i-1)$, $t1(i)$

TAMOM

yoki boshqacha takrorlanish tuzilmasi orqali yozsak:

o'tkaz 2, $t1(1)$

TAKRORLANSIN N-1 MARTA

o'tkaz $2 \cdot t1(i)$, $t1(i+1)$

TAMOM

Bu usulning samaradorligi avvalgi Bekning usuli bilan bir xil, murakkabligi esa Bekning usulidan bittaga ortiq, lekin asosiy yutuq algoritmda darajaga ko'tarish amali ishtirok etmayapti.

Algoritmikada bu kabi keyingi qadamni avvalgi qadamga bog'laydigan usul **iteratsiya usuli** deb ataladi.

Endi barcha tokchalardagi sonlarni S tokchaga yig'amiz:

bo'shat S

TAKRORLANSIN N MARTA

o'tkaz $S+t1(i)$, S

TAMOM

Algoritmni iteratsiya usuli yordamida yozilgan to'liq ko'rinishi quyidagicha:

o'tkaz 2, t1(1)

2 DAN N GACHA BAJAR

o'tkaz $2 \cdot t1(i-1)$, t1(i)

TAMOM

bo'shat S

TAKRORLANSIN N MARTA

o'tkaz $S+t1(i)$, S

TAMOM

Agar oxirgi algoritmning samaradorligini saqlagan holda murakkabligini kamaytirmoqchi bo'lsak, quyidagicha yozishimiz mumkin:

o'tkaz 2, t1(1)

o'tkaz t1(1), S

2 DAN N GACHA BAJAR

o'tkaz $2 \cdot t1(i-1)$, t1(i)

o'tkaz $S+t1(i)$, S

TAMOM

yoki boshqacha takrorlanish tuzilmasi orqali yozsak:

o'tkaz 2, t1(1)

o'tkaz t1(1), S

TAKRORLANSIN N-1 MARTA

o'tkaz $2 \cdot t1(i)$, t1(i+1)

o'tkaz $S+t1(i+1)$, S

TAMOM

Yig'ish uchun ishlatilayotgan S tokchani avvallari bo'shatib olardik. Bu ikkala algoritmida esa bo'shatib olganimiz yo'q. Buni sababini o'zingiz izohlang.

9.20-mashq

Masala yechimida keltirilgan algoritm to'g'ri ishlashini jadval yordamida tekshiring.

9.21-mashq

Saralovchi M berilgan N uchun 3 ning darajalarini t1(*) ning tokchalariga joylashtirib, keyin ularni S tokchaga yig'sin.

9.15-masala

Bekka ota-onasi shokoladlar berishdi. 1-kuni unga 5 ta shokolad berishdi. Keyingi har kuni avvalgisiga nisbatan 2 barobar ko'p shokolad berishdi. Bek bir kunda beriladigan shokoladlar soni K tadan oshguncha yemasdan yig'ib yurdi va ota-onasini mehmon qildi. U shokolad berishni boshlanganining nechanchi kuni ota-onasini mehmon qilgan?

Yechim. Bekka 1-kuni berilgan shokoladni $t_1(1)$ tokchaga, 2-kuni berilgan shokoladni $t_1(2)$ tokchaga va hokazo joylashtiriladi. Qaysi tartib raqamli tokchaga K tadan ortiq birinchi marta shokolad joylashtirilsa, shu tokchanning E_k dagi ifodasi masala javobi bo'ladi.

Masalaning yana bir tomoniga e'tiboringizni jalb qilmoqchimiz. Masala shartida tokchalar sonini yuqori chegarasi berilmagan. Yuqori chegara K ga bog'liq ravishda o'zgaradi.

Haqiqatan:

K ning qiymati	Kunlarga mos shokoladlar soni							Yuqori chegaraning qiymati
	1	2	3	4	5	6	7	
1	5							1
7	5	10						2
9	5	10						2
10	5	10	20					3
21	5	10	20	40				4
39	5	10	20	40				4
300	5	10	20	40	80	160	320	7

Demak, K qanchalik kattalashsa N ham shunga bog'liq ravishda ortishi mumkin ekan. Yuqori chegara aniq bo'lmaganda takrorlanish qanday tashkil etiladi?

Bu savolga TOKI – BAJAR tuzilmasi javob beradi:

o'tkaz 1, E_k

o'tkaz 5, $t_1(E_k)$

TOKI $t_1(E_k) \leq K$ BAJAR

o'tkaz E_k+1 , E_k

o'tkaz $2 \cdot t_1(E_k-1)$, $t_1(E_k)$

TAMOM

Algoritmi ishlashini $K=99$ da jadval yordamida ko'rib chiqamiz:

Ko'rsatma	Ek	$t1(*)$	Shart qiymati	Ek+1	Ek-1
o'tkaz 1, Ek	1	-	-	-	-
o'tkaz 5, $t1(Ek)$	1	$t1(1)=5$	-	-	-
$5=t1(Ek) \leq K=99$	1	-	ROST	2	1
o'tkaz Ek+1, Ek	2	-	-	-	-
o'tkaz $2 \cdot t1(Ek-1)$, $t1(Ek)$	2	$t1(2)=2 \cdot t1(1)=2 \cdot 5=10$	-	-	-
$10=t1(Ek) \leq K=99$	2	-	ROST	3	2
o'tkaz Ek+1, Ek	3	-	-	-	-
o'tkaz $2 \cdot t1(Ek-1)$, $t1(Ek)$	3	$t1(3)=2 \cdot t1(2)=2 \cdot 10=20$	-	-	-
$20=t1(Ek) \leq K=99$	3	-	ROST	4	3
o'tkaz Ek+1, Ek	4	-	-	-	-
o'tkaz $2 \cdot t1(Ek-1)$, $t1(Ek)$	4	$t1(4)=2 \cdot t1(3)=2 \cdot 20=40$	-	-	-
$40=t1(Ek) \leq K=99$	4	-	ROST	5	4
o'tkaz Ek+1, Ek	5	-	-	-	-
o'tkaz $2 \cdot t1(Ek-1)$, $t1(Ek)$	5	$t1(5)=2 \cdot t1(4)=2 \cdot 40=80$	-	-	-
$80=t1(Ek) \leq K=99$	5	-	ROST	6	5
o'tkaz Ek+1, Ek	6	-	-	-	-
o'tkaz $2 \cdot t1(Ek-1)$, $t1(Ek)$	6	$t1(6)=2 \cdot t1(5)=2 \cdot 80=160$	-	-	-
$160=t1(Ek) \leq K=99$	6	-	YOLG'ON	To'xtaydi	

Demak, $K=99$ bo'lganda Bek ota-onasini 6-kun mehmon qilgan.

9.23-mashq

Saralovchi M uchun Bek ota-onasini mehmon qilganda nechta shokoladi borligini aniqlovchi algoritim tuzing.

Qo'shimcha masalalar

Keling, bu bo'limda Saralovchi M ni biz boshqaramiz. Quyidagicha talab bilan $t1(*)$ ning tokchalarini to'ldiramiz:

$$t1(1)=0; t1(2)=1;$$

$$t1(3)=t1(1)+t1(2); t1(4)=t1(2)+t1(3); \dots,$$

ya'ni, keyingi har bir tokchadagi son oldingi ikkita tokchadagi sonlarning yig'indisiga teng. Bu usulda hosil qilinadigan sonlarni **Fibonacci sonlari** deb atashadi.

9.16-masala

Saralovchi M birinchi N ta Fibonacci sonini hosil qilsin.

Yechim. Fibonacci sonlarining ta'rifiga ko'ra algoritm tuzish yetarli:

o'tkaz 0, $t_1(1)$

o'tkaz 1, $t_1(2)$

3 DAN N GACHA BAJAR

o'tkaz $t_1(i-2)+t_1(i-1)$, $t_1(i)$

TAMOM

9.24-mashq

Saralovchi M 1 dan 50 gacha bo'lgan sonlar ichida nechta Fibonacci soni borligini aniqlasin.

Quyidagilarni eslatib o'tamiz:

- H natural son N natural sonining **bo'luvchisi** deb ataladi, agar shunday G natural son topilsaki, $N=G \cdot H$ (yoki $N/H=G$) shart bajarilsa.
- H natural son natural N va K sonlarining **umumiy bo'luvchisi** deb ataladi, agar H son N ning ham, K ning ham bo'luvchisi bo'lsa.
- H natural son natural N va K sonlarining **eng katta umumiy bo'luvchisi** deb ataladi, agar H son N va K sonlarining umumiy bo'luvchilari ichida eng kattasi bo'lsa.
- H natural son N natural sonining **karralisi** deb ataladi, agar shunday G natural son topilsaki, $H=G \cdot N$ (yoki $H/N=G$) shart bajarilsa.
- H natural son natural N va K sonlarining **umumiy karralisi** deb ataladi, agar H son N ning ham, K ning ham karralisi bo'lsa.
- H natural son natural N va K sonlarining **eng kichik umumiy karralisi** deb ataladi, agar H son N va K sonlarining umumiy karralilari ichida eng kichigi bo'lsa.

9.17-masala

Saralovchi M ikkita N va K sonlarni eng katta umumiy bo'luvchisi $EKUB(N, K)$ ni topsin.

Yechim. 1-usul. Bu masalani yechishni ta'rif bo'yicha bajarishga harakat qilish mumkin, ya'ni sonlar teng bo'lsa $EKUB=N$, aks holda ikkala sonni bir chekkadan natural sonlarga bo'lib boraveramiz va har qadamda ikkala sonning bo'luvchisini $EKUB$ deb olaveramiz, bu holda jarayon bo'luvchi bo'linuvchilardan kichigiga teng bo'lishi bilan to'xtatiladi. Lekin bu usulda bir muammo bor: bo'linish shartini Saralovchi M tushunadigan tarzda qanday yozish mumkin. Biz biladigan bo'linish shartlari (qoldiq nolga tengligi $- N \bmod i = 0$ yoki butun bo'linish sharti $- [N/i] = N/i$, bu yerda $[a]$ $- a$ sonini butun qismi) Saralovchi M uchun **tushunarli emas.**

Bo'linish degan amalni boshqa mazmuniga e'tibor beramiz: agar D soni H soniga bo'linsa, u holda shunday G son mavjudki, D dan G marta H ni ayirsak 0 hosil bo'ladi. Haqiqatan, agar D soni H soniga bo'linsa, u holda

$$D = G \cdot H = \underbrace{H + H + H + H + \dots + H}_{G \text{ ta}}$$

yoki

$$D - \underbrace{(H + H + H + H + \dots + H)}_{G \text{ ta}} = 0$$

Shuni e'tiborga olib N sonini i ga bo'lgandagi qoldiqni topish uchun quyidagi algoritmnı tuzamiz:

o'tkaz N , Qoldiq N

TOKI Qoldiq $N \geq i$ BAJAR

o'tkaz Qoldiq $N - i$, Qoldiq N

TAMOM

Endi bu algoritmnı qo'llab masalani hal eta olamiz:

AGAR $N = K$

U HOLDA

o'tkaz N , $EKUB$ { $EKUB$ aniqlandi}

AKS HOLDA

AGAR $N > K$ {sonlardan kichigini aniqlash}

U HOLDA

o'tkaz N , S

AKS HOLDA

o'tkaz K , S

TAMOM

o'tkaz 1 , $EKUB$

{har qanday son 1 ga bo'linadi}

2 DAN S GACHA BAJAR

{bo'luvchilar qadami}

o'tkaz N, QoldiqN

TOKI QoldiqN \geq i BAJAR

o'tkaz QoldiqN - i, QoldiqN

TAMOM

o'tkaz K, QoldiqK

TOKI QoldiqK \geq i BAJAR

o'tkaz QoldiqK - i, QoldiqK

TAMOM

AGAR QoldiqN = 0 VA QoldiqK = 0

U HOLDA

o'tkaz i, EKUB

TAMOM

TAMOM

TAMOM

Bu yerda S sonlardan kichigini aniqlaydi. Lekin algoritm juda uzun va samaradorligi past. EKUB topish muammo bo'lib qoldiku!

2-usul. Lekin muammoni eramizdan avvalgi III asrda yashagan matematik Evklid osongina hal qilib qo'ygan ekan. Uning algoritmi deyarli quyidagicha ifodalanadi:

Evklid algoritmi: agar N va K sonlar teng bo'lsa, u holda EKUB=N; aks holda kattasidan kichigini ayirib kattasini o'rniga qabul qilamiz va algoritmni yana boshidan boshlaymiz.

Evklid algoritmini misollarda ko'ramiz:

N	K	Shart	Qiymati	Shart	ROST, ya'ni u holda	YOLG'ON, ya'ni aks holda
15	15	15<>15	YOLG'ON	EKUB=15		
N	K	Shart	Qiymati	Shart	ROST, ya'ni u holda	YOLG'ON, ya'ni aks holda
15	12	15<>12	ROST	15>12	N=15-12=3	-
3	12	3<>12	ROST	3>12	-	K=12-3=9
3	9	3<>9	ROST	3>9	-	K=9-3=6
3	6	3<>6	ROST	3>6	-	K=6-3=3
3	3	3=3	YOLG'ON	EKUB=3		

N	K	Shart	Qiymati	Shart	ROST, ya'ni u holda	YOLG'ON, ya'ni aks holda
15	4	15<>16	ROST	15>4	N=15-4=11	-
11	4	11<>4	ROST	11>4	N=11-4=7	-
7	4	7<>4	ROST	7>4	N=7-4=3	-
3	4	3<>4	ROST	3>4	-	K=4-3=1
3	1	3<>1	ROST	3>1	N=3-1=2	-
2	1	2<>1	ROST	2>1	N=2-1=1	-
1	1	1=1	YOLG'ON	EKUB=1		

Mana bu boshqa gap. Faqat takrorlanish qadamini oldindan bilmaymiz, shuning uchun TOKI – BAJAR tuzilmasidan foydalanib tuzilgan Saralovchi M tushunadigan algoritm quyida-gicha bo'ladi:

TOKI N <> K BAJAR
AGAR N>K
U HOLDA
o'tkaz N-K, N
AKS HOLDA
o'tkaz K-N, K
TAMOM
TAMOM
o'tkaz N, EKUB

Jadvalda va algoritmda takrorlash shartini Evklid algoritmidagi asosiy shartdan farqli teng emaslik sharti kabi yozilishi dasturni murakkablik darajasini kamaytiradi.

9.24-mashq

Evklid algoritmini algoritmdagi teng emaslik shartini tenglik shartiga almashtirib tuzing.

9.18-masala

Saralovchi M ikkita N va K sonlarning eng kichik umumiy karralisi EKUK(N, K) ni topsin.

Yechim. EKUB ning ta'rif bo'yicha topish algoritmini ko'rdingiz. EKUK ni ham ta'rif bo'yicha topish ham ancha murakkab. Shuning uchun, EKUB va EKUK ni quyidagi bog'lanishidan foydalanish maqsadga muvofiq:

$$EKUB(N, K) \cdot EKUK(N, K) = N \cdot K$$

Bu bog'lanishdan quyidagini hosil qilamiz:

$$EKUK(N, K) = N \cdot K / EKUB(N, K)$$

EKUB ni topishni Evklid algoritmidan foydalanib masalani hal etamiz:

o'tkaz $N \cdot K, S$
TOKI $N <> K$ BAJAR
AGAR $N > K$
U HOLDA
o'tkaz $N-K, N$
AKS HOLDA
o'tkaz $K-N, K$
TAMOM
TAMOM
o'tkaz $N, EKUB$
o'tkaz $S / EKUB, EKUK$

Algoritm bajarilish jarayonida N va K ning qiymati kamayib boradi, shuning uchun ularni boshlang'ich qiymatiga mos ko'paytmani S tokchada saqlab turduk.

9.25-mashq

Bo'yi 96 m, eni esa 88 m ga teng to'g'ri to'rtburchak shaklidagi kartonni teng kvadratlarga ajratishmoqchi. Shu ma'noda eng katta tomonli kvadratning tomoni uzunligi necha m ga teng bo'ladi?

Birdan farqli natural sonni faqat ikkita bo'luvchiga ega bo'lsa: o'zi va bir, u tub son deyiladi. Tub sonlarning birinchisi 2 ga teng.

Buni qarangki, 2 yakkayu-yagona juft tub son bo'lar ekan. Boshqa har qanday juft son tub bo'la olmaydi, chunki uning hech bo'lmaganda yana bir uchinchi bo'luvchisi 2 bor (o'zi va birdan tashqari). Quyida 100 dan kichik tub sonlar keltirilgan:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.

N va K o'zaro tub sonlar deyiladi, agar $EKUB(N, K)=1$ bo'lsa. Ma'lumki, har qanday ikkita tub son o'zaro tub bo'ladi. Shu yerda Suvchi uchun bir xossani aytib o'tmoqchimiz:

A litrli va B litrli idishlar bor. Suvchi faqat A va B dan oshmaydigan EKUB(A, B) ga karrali har qanday hajmdagi suvni o'lchab olishi mumkin.

Ya'ni, masalan, $A=2$ va $B=8$ bo'lsa, $EKUB(2, 8)=2$ va shuning uchun Suvchi 2, 4, 6, 8 litr suvni o'lchab olishi mumkin, 1, 3, 5, 7 litr suvni o'lchab ololmaydi.

9.19-masala

Saralovchi M berilgan N son tub bo'lsa, T tokchaga 1 ni, aks holda 0 ni o'tkazsin.

Yechim. Sonni tub ekanligini tekshirish uchun uni bo'luvchilar soni 2 ga teng yoki teng emasligini tekshirish kifoya. Buning uchun bo'linish shartini qoldiq ko'rinishida tekshiramiz:

o'tkaz 0, S

o'tkaz 0, T

1 DAN N GACHA BAJAR

{bo'luvchilar qadami}

o'tkaz N, QoldiqN

TOKI QoldiqN \geq i BAJAR

{qoldiqni aniqlash}

o'tkaz QoldiqN - i, QoldiqN

TAMOM

AGAR QoldiqN=0

{bo'linish alomatini tekshirish}

U HOLDA

o'tkaz S+1, S

TAMOM

TAMOM

AGAR S=2

{tub bo'lish alomati}

U HOLDA

o'tkaz 1, T

TAMOM

9.20-masala

Saralovchi M 100 dan kichik barcha tub sonlarni $t1(*)$ ning tokchalariga yozib chiqsin.

Yo'llanma. Tub son bo'lish xossasini tekshirishni bilasiz. Endi uni 1 dan 99 gacha bo'lgan sonlar uchun takrorlash yetarli. Lekin 100 dan kichik tub sonlarning nechtaligi noma'lum, shuning uchun $t1(*)$ ning tokchalariga murojaat qilishda TOKI – BAJAR takrorlash tuzilmasidan foydalanish maqsadga muvofiq.

9.21-masala

Saralovchi M 100 dan N va K sonlarining tub bo'luvchilarini $t1(*)$ ning tokchalariga yozib chiqsin.

Nazorat savollari va topshiriqlar

1. Saralovchi M imkoniyatlarini avvalgi saralovchilar bilan taqqoslab bering.
2. Saralovchi M ning Zt va Ek qurilmalari qanday imkoniyatlar beradi?
3. Qanday sonlar polindrom sonlar deyiladi? Misollar keltiring.
4. Iteratsiya usulini misollar orqali izohlang.
5. Qanday sonlar Fibonacci sonlari deb ataladi?
6. Qadamlarning yuqori chegarasi noma'lum bo'lsa, takrorlash qanday tashkil etiladi?
7. Qoldiq hisoblash algoritmini misollarda izohlang.
8. Evklid algoritmini misollarda ko'rsatib bering.
9. Ikki sonning EKUKini topish algoritmini izohlang.
10. Tub son ta'rifini misollar orqali izohlang.
11. Berilgan son tubligini tekshirish algoritmini izohlang.
12. Masalalarning algoritmini blok-sxema ko'rinishida tasvirlang.
13. Bobdagi barcha mashqlarni bajaring.

Qo'shimcha masalalar

S-M-9.1. Saralovchi M N ta tokchali $t1(*)$ tokchalaridagi buyumlarni teskari tartibda joylashtirsin, ya'ni 1-tokchadagi buyumlarni N -tokchadagi buyumlar bilan, 2-tokchadagi buyumlarni $(N-1)$ -tokchadagi buyumlar, 3-tokchadagi buyumlarni $(N-2)$ -tokchadagi buyumlar bilan o'rnini almashtirsin.

S-M-9.2. N ta tokchali $t1(*)$ tokchalarida yoki 1 ta kub yoki 2 ta kub yoki 3 ta kub bor. Saralovchi M $t1(*)$ tokchalaridagi kublarni shunday joylashtirsin, avval faqat 1 ta kubli tokchalar, keyin faqat 2 ta kubli tokchalar, oxirida faqat 3 ta kubli tokchalar hosil bo'lsin.

S-M-9.3. Saralovchi M N ta tokchali $t1(*)$ ni buyum miqdori eng ko'p tokchasini buyumlarini N -tokchaga buyum miqdori eng kam bo'lgan tokchasini buyumlarini 1-tokchaga o'tkazsin.

S-M-9.4. Saralovchi M N ta tokchali $t1(*)$ ni buyum miqdori eng ko'p tokchasini buyumlarini barcha tokchalarga o'tkazsin.

S-M-9.5. Saralovchi M N ta tokchali $t1(*)$ ni juft tartib raqamli tokchalardagi buyum miqdori eng ko'p tokchasini toq tartib raqamli

tokchalardagi buyum miqdori eng kam bo'lgan tokchasiga qo'shini Zt tokchaga o'tkazsin.

S-M-9.6. Saralovchi M N ta tokchali $t1(*)$ ni tokchalarini toq sonlar bilan o'sish tar-tibida to'ldirsin.

S-M-9.7. Saralovchi M N ta tokchali $t1(*)$ ni tokchalarini juft sonlar bilan kamayish tartibida to'ldirsin.

S-M-9.8. Saralovchi M N ta tokchali $t1(*)$ ni tokchalariga quyidagilarni tartib bilan joylasin:

$$1\cdot2, 2\cdot3, 3\cdot4, \dots, N\cdot(N+1)$$

va ularni yig'indisini S tokchaga yig'sin.

S-M-9.9. Saralovchi M N ta tokchali $t1(*)$ ni tokchalariga quyidagilarni tartib bilan joylasin:

$$1\cdot2\cdot3, 2\cdot3\cdot4, 3\cdot4\cdot5, \dots, N\cdot(N+1)\cdot(N+2)$$

va ularni yig'indisini S tokchaga yig'sin.

S-M-9.10. Saralovchi M N ta tokchali $t1(*)$ ni tokchalariga quyidagilarni tartib bilan joylasin:

$$1\cdot2\cdot3, 4\cdot5\cdot6, 7\cdot8\cdot9, \dots, (3\cdot N-2)\cdot(3\cdot N-1)\cdot(3\cdot N)$$

va ularni yig'indisini S tokchaga yig'sin.

S-M-9.11. Saralovchi M N ta tokchali $t1(*)$ ni tokchalariga quyidagilarni tartib bilan joylasin:

$$(1\cdot2)^2, (2\cdot3)^2, (3\cdot4)^2, \dots, (N\cdot(N+1))^2$$

va ularni yig'indisini S tokchaga yig'sin.

S-M-9.12. Saralovchi M N ta tokchali $t1(*)$ ni tokchalariga quyidagilarni tartib bilan joylasin:

$$(1\cdot2), (2\cdot3\cdot4), (3\cdot4\cdot5\cdot6), \dots$$

va ularni yig'indisini S tokchaga yig'sin.

S-M-9.13. Saralovchi M N ta tokchali $t1(*)$ va $t2(*)$ ni mos tartib raqamli tokchalarini yig'indisini, $t3(*)$ qavatning mos tartib raqamli tokchalariga o'tkazsin, ya'ni $t1(i) + t2(i)$ ni $t3(i)$ ga o'tkazsin.

S-M-9.14. Saralovchi M N ta tokchali $t1(*)$ va $t2(*)$ ni mos tartib raqamli tokchalarini taqqoslab, agar $t1(i) > t2(i)$ shart bajarilsa $t3(i)$ ga 1 ni, aks holda $t3(i)$ ga 0 ni o'tkazsin.

S-M-9.15. Saralovchi M N ta tokchali $t1(*)$ va $t2(*)$ ni mos tartib raqamli tokchalarini taqqoslab, $t1(i) > t2(i)$ shart bajarilgan tokchalar sonini hisoblasin.

S-M-9.16. Saralovchi M N ta tokchali $t1(*)$ va $t2(*)$ ni tokchalaridagi sonlarni $2 \cdot N$ ta tokchali $t3(*)$ qavatga quyidagicha tartib bilan joylasin:

$$t1(1), t2(1), t1(2), t2(2), t1(3), t2(3), \dots, t1(N-1), t2(N-1), \\ t1(N), t2(N).$$

S-M-9.17. Bekka ota-onasi shokoladlar berishdi. 1-kuni unga 14 ta shokolad berishdi. Keyingi kuni avvalgisiga nisbatan 2 barobar ko'p, uchinchi kun avvalgisiga nisbatan 3 barobar ko'p, to'rtinchi kun avvalgisiga nisbatan 4 barobar ko'p, va hokazo, shokolad berishdi. Bek bir kunda beriladigan shokoladlar soni K tadan oshguncha yemasdan yig'ib yurdi va ota-onasini mehmon qildi. U shokolad berishni boshlanganini nechanchi kuni ota-onasini mehmon qilgan?

S-M-9.18. Fibonacci sonlarini S tokchaga yiqqanda qaysi tartib raqamli tokchadagi sonni qo'shganda yig'indi berilgan K sonidan ortib ketadi.

S-M-9.19. Orasidagi masofa 300 km bo'lgan A va B shaharlardan bir vaqtning o'zida bir-biriga qarab ikki velosipedchi soatiga 50 km tezlik bilan yo'lga chiqdi. Shu zahoti 1-velosipedchining peshonasidagi pashsha soatiga 100 km tezlik bilan 2-velosipedchiga qarab uchdi. Pashsha 2-velosipedchining peshonasiga urilib ortga qaytdi. Va bu kabi urilish ikki velosipedchi uchrashguncha davom etdi. Saralovchi M velosipedchilar uchrashguncha pashshaning bosib o'tgan masofasini aniqlasin.

S-M-9.20. Saralovchi M 100 dan kichik Fibonacci sonlaridan nechitasi tub bo'lishini aniqlasin.

S-M-9.21. Saralovchi M 100 dan N va K sonlarini bo'luvchilarini $t1(*)$ ni tokchalariga yozib chiqsin.

Tokchalar uchun ba'zi chegaralashlar

Shu vaqtgacha Saralovchi M ning tokchalariga istalgancha buyum qo'ygan edik, lekin «tokcha ko'tara olmasa-chi yoki tokchaga sig'masa-chi» degan savolni berganimiz yo'q edi. Hayotda har qanday tokcha qandaydir ma'noda chegaralangan bo'ladi. Ba'zan tokcha bir necha xil chegaralashga ega bo'ladi, ya'ni, masalan, shkaf tokchasi ham hajm jihatidan (5x6x7 metr o'lchamdagi quti qaysi shkaf tokchasiga sig'adi), ham og'irlik ko'tarish jihatidan (500 kg yukni qaysi shkaf tokchasi ko'tara oladi) ham buyum xususiyati jihatidan (suv yoki gazni shkafingiz tokchasida maxsus idishlarsiz saqlay olasizmi) chegaralangan. Bu chegaralashlar har xil tokchaga turli xil ko'rinishda bo'lgani uchun ularni Saralovchi M ni emas **tokchani xususiyati** deb ataymiz. Tokchani imkoniyati chegarasini bildirish uchun **yuqori chegarani** aniqlaymiz.

Oddiy bir misol qaraymiz. Faqat ikkita raqami bor Saralovchi M ning tokchalari faqat 0 yoki 1 birlik (masalan, 0 va 1 metr yoki 0 va 1 kub metr yoki 0 va 1 kg) o'lchamdagi buyum miqdorini saqlay olsin. U holda 1 birlikdan oshib ketsachi? Unda nima qilamiz?

Buni yo'li oson ekan, shunday qoida kiritamiz:

Agar tokchadagi buyum miqdori biror amaldan keyin yuqori chegaradan ortib ketsa, u holda bu tokchaga buyum miqdorini yuqori chegaraga bo'lgandagi qoldig'i o'tkaziladi va keyingi tartib raqamli tokchaga bo'lganda hosil bo'lgan butun qismga mos birlikda buyum miqdori qo'shiladi.

Bu Sizga yangilik emas, bu kabi ishni o'nlik sanoq sistemasida biror sonni ikkinchi songa qo'shganda yoki ko'paytirganda bajar-gansiz: biror xonadagi yig'indi yoki ko'paytmaning 10 dan kichik qismi (raqam) shu xonada qoldiriladi, dildagi (o'nga bo'lganda hosil bo'lgan butun) qism (raqam) keyingisiga qo'shiladi.

Yuqoridagi misolda tokchaga 0 yoki 1 ta sig'adi. Agar tokcha to'lgan bo'lsa, ya'ni unda 1 bo'lsa, 1 ni qo'shsak sig'maydi va bu

tokcha 0 yozilib keyingisiga 1 o'tkaziladi. Quyidagi jadvalda ikkita raqamli tokchalarda qo'shish qanday olib borilishi ko'rsatilgan:

t1(1)	t1(2)	t1(3)	t1(4)	Amal
0				+1
1				+1
0	1			+1
1	1			+1
0	0	1		+1
1	0	1		+1
0	1	1		+1
1	1	1		+1
0	0	0	1	

Agar tokchalarga 3 ta raqam 0, 1, 2 sig'sa:

t1(1)	t1(2)	t1(3)	Amal
0			+1
1			+1
2			+1
0	1		+1
1	1		+1
2	1		+1
0	2		+1
1	2		
2	2		
0	0	1	

Ba'zan, quyi chegara haqida ham so'z yuritish mumkin, bu holda ayirish va bo'lish amallari yoki yana boshqa amallar haqida mulohaza qilish mumkin. Demak, tokchalarni xususiyatini belgilab turli maqsadlarda foydalanish mumkin ekan.

Dasturchilar shunday deyishadi: **algoritm tuzish yillab o'rganiladi, dasturlash tili esa 1 haftada.** Ushbu qo'llanmada ham asosiy e'tibor algoritm tuzishni o'rgatishga qaratilganini tushungan bo'lsangiz kerak.

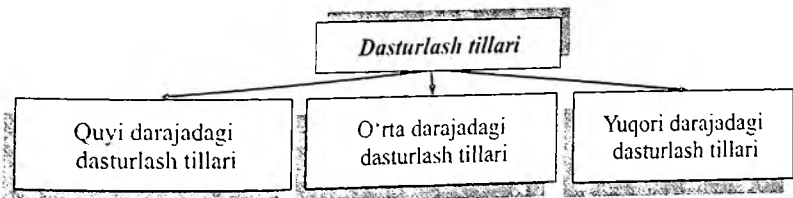
Dastur va dasturlash tillari

Ma'lumki, kompyuter texnikasidan samarali foydalanish ikki qism – texnik va dasturiy ta'minotning uzviyligini talab etadi. Bu uzviylik kompyuter texnik ta'minotining jadal sur'atlar bilan takomillashib borishiga mos dasturiy ta'minotni ham keskin sur'atlar bilan rivojlanishiga sabab bo'ladi, va aksincha. Buning sababi ma'lum, mos dasturiy ta'minotsiz har qanday kompyuter «qimmatbaho o'yinchoq» bo'lib qolaveradi.

Ma'lumki, kompyuterda biror masalani hal qilish uchun avval uning qandaydir nusxasi olinadi va algoritmi tuziladi, so'ng mazkur algoritm ma'lum bir qonun-qoidalar asosida kompyuter tushunadigan ko'rsatma va buyruqlar shaklida yoziladi. Hosil bo'lgan matn kompyuter tilida yozilgach, **dastur** deb ataladi. Demak, **dastur** – biror masalani yechish uchun kompyuter bajarishi mumkin bo'lgan ko'rsatmalarning izchil tartibi ekan.

Kompyuter uchun dastur tuzish jarayoni **dasturlash** va dastur tuzadigan kishi **dasturchi** deyiladi. Kompyuter tushunadigan «til» esa **dasturlash tili** deb ataladi.

Dasturlash tillarini shartli ravishda quyidagi uch guruhga ajratish mumkin:



Dasturlash tillari tarixidan. Dasturlash tillari, asosan, ikkinchi jahon urushidan keyin yaratila boshlandi. Ammo uning boshlanish tarixi ancha olis yillarga borib taqaladi.

Arxeologik qazilmalarda topilgan sopol taxtachada bundan 3800 yil oldin (eramizdan avvalgi 1800 yillar) Bobilda foiz bilan bog'liq murakkab amallar algoritmi keltirilgan. Unda aniq masala ishlangan bo'lib, agar bug'doy hosili yiliga 20% dan oshib borsa, uning miqdori ikki marta o'sishi uchun necha yil va oy kerak bo'lish algoritmi tuzilgan.





XIX asr fransuz kashfiyotchisi **Jozef Mari Jakkard** 1804 yilda yupqa mato ishlab chiqish jarayonida to'quv dastgohlari uchun perfokartani eslatuvchi tasma ishlatgan va shu bilan perfokartaga asos solgan edi.

Charlz Bebbij

1836 yilda ingliz olimi **Charlz Bebbij** hozirgi kompyuterlarning bevosita ajdodi bo'lmish analitik mashina ishlab chiqishga kirishdi va bu masalani nazariy hal qildi. Bu mashinaning asosiy xususiyati uning dastur asosida ishlashi va hisob-kitob natijalarini

«eslab» qolishida edi.

1843 yilda ingliz matematigi Ogasta **Ada Bayron** (Lavleys) – shoir lord Bayronning qizi analitik mashina buyruqlar asosida ishlashi lozimligini ta'kidladi. U berilgan shartlar bajarilmagunga qadar qadamlar ketma-ketligini ta'minlovchi buyruqlarni yozdi. Ana shu holat bilan u dasturlash tiliga asos soldi. Mazkur va boshqa kashfiyotlar kompyuter yaratilgach, ularni ishlatish uchun zarur bo'lgan til yaratilishini talab etdi.



Ada Bayron

Quyi darajadagi dasturlash tillari kompyuter qurilmalari bilan bevosita bog'liq bo'lib, buyruqlar maxsus raqamlar (kodlar) yordamida yoziladi. Bu kabi buyruqlardan tashkil topgan dasturlar katta hajmli bo'lib, ularni tahrir qilish ancha mushkul ish hisoblanadi. Dastlabki elektron hisoblash mashinalarida («ENIAC», «MESM» va boshqalar) masalalarni yechish uchun ana shunday buyruqlar yordamida dasturlar tuzilgan.

Misol tariqasida M-20 rusumidagi elektron hisoblash mashinasida qo'llanilgan tilda tuzilgan dasturni (dastur doira yuzini hisoblash amallarini o'z ichiga olgan) izoh bilan keltiramiz:

Buyruqning kodi	Buyruqqa izoh
01 022	R radiusning qiymati jamlagichga yuboriladi
20 000	Jamlagichdagi qiymat (R)ni bosmaga chiqarish
05 022	Jamlagichdagi R ning qiymati o'z-o'ziga ko'paytiriladi va natija yana jamlagichga yoziladi
05 020	020—021-yacheykalaridagi π soniga jamlagichdagi qiymat (R^2) ko'paytiriladi

20 000	Olingan natija, ya'ni doira yuzasining qiymati bosmaga chiqariladi
045 00 000	Tamom (Stop)

Ko'rinib turibdiki, bu tilda dastur tuzish ancha mashaqqatli ekan. Buning asosiy qiyinchiligi – bir tomondan buyruqlarning raqamlar yordamida ifodalanishi bo'lsa, ikkinchi tomondan dasturchidan har bir amalni bajarilishida jamlagichdagi sonli qiymatning qaysi o'zgaruvchiga tegishligini va boshqa o'zgaruvchilarning qiymatlari qaysi adresda joylashganligini bilish talab etiladi.

Dastur tuzishni osonlashtirish maqsadida inson tiliga yaqin bo'lgan buyruqlar tizimini tuzish va qo'llash masalasi qo'yildi hamda hal etildi. Bu kabi dasturlash tillari **o'rta darajadagi dasturlash tillari** (ba'zan **assemblerlar**) deb yuritila boshlandi. Bunday tillarga **AVTOKOD-BEMSH, AVTOKOD-MADLEN** va boshqalar kiradi. Ular **BESM-6, Minsk-22, Minsk-32, IBM-360** elektron hisoblash mashinalarida ishlatildi. Masalan,

ST 5, BSUM

ifodada 5 raqami BSUM deb nomlangan yacheykaga joylashtirilsin (**ST-store** – joylashtirish), degan buyruq berilgan.



Assembler tillarida buyruqlar qisqartirilgan so'zlar yoki so'zlar majmuidan iborat bo'lib, ular **mнемokodlar** deb ham yuritiladi.

Ta'kidlash joizki, dasturlash davomida yo'l qo'yilgan biror xato salbiy natijalarga olib kelishi ham mumkin.

1981 - yil 10 - aprel. Amerika Qo'shma Shtatlarining Kanaravel kosmodromidan birinchi bor ko'p marta qo'llanilishga mo'ljallangan «Shatll» rusumidagi kosmik kemani uchishga tayyorlash vaqtida uni boshqarishga mo'ljallangan barcha kompyuterlar xatolik yuzaga kelganligi to'g'risida ma'lumot berdi. Bu kabi xatolikni kema bortida o'rnatilgan kompyuter ishini sinxron ravishda takrorlovchi boshqaruv Markazidagi kompyuter ham ko'rsatdi. Bu holatda kosmik kemani fazoga uchirish xavfli, albatta. Kemasidagi barcha jarayonlar kompyuter yordamida boshqarilishga mo'ljallangan bo'lib, ulardagi dastur 500 mingdan ziyod turli buyruqlarni o'z ichiga olgan edi. Mutaxassislar tomonidan parvozni boshqarish uchun mo'ljallangan o'ndan ziyod sinxron ravishda ishlovchi kompyuterlarning amal bajarishi o'rtasidagi vaqtning farqi 30 mks ekanligi aniqlandi hamda buning, umuman olganda, xavfli emasligini hisobga olib, ikki kunga kechiktirilgan holda kema parvozi amalga oshirildi.

Yuqori darajali dasturlash tillaridagi ko'rsatmalar inson tiliga yaqin bo'lgan so'zlar majmuidan iborat. Ular yordamida amallarni bajarish quyi darajadagilaridan ko'ra yengil bo'lib, biror maxsus ko'rsatma bo'lmasa, dasturchidan adreslar, qurilmalar bilan bevosita bog'liq axborotlarni bilish talab etilmaydi. Bu tilda tuzilgan dasturlarni **translyatorlar** deb nomlanuvchi maxsus dasturlar kompyuterlar bajara olishi uchun raqamli ko'rinishga o'tkazib beradi.

Keyingi yillarda juda ko'p yuqori darajadagi dasturlash tillari yaratilgan bo'lib, ular qatoriga **Paskal, dBase, Ada, KARAT, C++, Delphi, Visual Basic** va boshqa tillarni qo'shish mumkin. Hozirgi kunda yaratilayotgan dasturlash tillari biror yo'nalishdagi masalalarni hal qilishga mo'ljallangandir.

Quyidagi jadvalda dasturlash tili rivojlanishi tarixidan qisqacha ma'lumot berilgan.

Dasturlash tili	Yaratilgan yili
Plankalkyul	1946
Qisqa kod	1949
Assembler «Edsak»	1950
AO	1950
Avtokod «Madlen»	1953
Tezkor kodlash	1955
A-2, Flou-metik	1956
IPL-1, Mat-metik	1957
Fortran	1958
Algol 58	1959
APT, LISP, Kobol, Algol-60	1960

Dasturlash tili	Yaratilgan yili
PL/1, Beysik	1964
Algol W	1965
Logo	1967
Algol 68	1968
APL	1969
Paskal	1970
Fort	1971
Prolog, Si	1972
Ada	1972
Smalltalk	1980

Shuni ham ta'kidlash kerakki, turli rusumdagi kompyuterlar uchun dasturlash tilining ularga moslashtirilgan naqllari ishlab chiqilgan bo'lib, ular bu tilning boshlang'ich naqlidan farq qilishi mumkin.

Yuqori darajadagi dastlabki dasturlash tili "**Plankalkyul**" deb nomlanib, u 1946 - yilda olmon olimi **Konrad Suzi** tomonidan tuzildi. Bu til o'z vaqtida ma'lum sabablarga (jumladan, ikkinchi jahon urushi oqibatlariga) ko'ra keng jamoatchilikka tanish emas edi. U 1972 yildan amalda qo'llanila boshlandi.

1949 yilda amerikalik **Jon Mouchli** dasturlashda 8 ta va 10 ta raqamli sanoq sistemalaridan foydalanmaslik taklifi bilan chiqdi. Ana shunga asoslangan dasturlash tili "Qisqacha kod" nomi bilan **Greys Holler** tomonidan yaratildi va dastlabki EHM larda ishlatildi.

Ijrochilar va dasturlash tillari

Avvalgi boblarda turli ijrochilar bilan tanishdik va algoritmlar tuzdik. Endi ba'zi dasturlash tillarida nomlar, ko'rsatmalar, tuzilmalar va boshqalar qanday bo'lishini ko'rib chiqamiz. Biz ko'rmoqchi bo'lgan dasturlash tillarida o'xshashliklar ko'p. Masalan, ularning alifbosi quyidagi asosiy qismlardan iborat:

Lotin alifbosining 26 ta harfi: Aa, Bb, Cc, Dd, Ee, Ff, Gg, Hh, Ii, Jj, Kk, Ll, Mm, Nn, Oo, Pp, Qq, Rr, Ss, Tt, Uu, Vv, Ww, Xx, Yy, Zz ;

O'nta arab raqami: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ;

Arifmetik amal belgilari: + (qo'shish), - (ayirish), * (ko'paytirish), / (bo'lish);

Munosabat belgilari: =(teng), <>(teng emas), <(kichik), <=(katta emas), >(katta), >=(kichik emas);

Maxsus belgilar: . (nuqta), , (vergul), ; (nuqtali vergul), ' (apostrof), « (qo'shtirnoq), ! (undov), ? (so'roq), % (foiz), \$ (dollar belgisi), @ (tijorat belgisi), & (ampersand), (bo'shliq, ekranda tasvirlanmaydi), (,), {, }, [,] (turli qavslar);

Mantiqiy amallar:

AND («VA» – mantiqiy ko'paytirish amali),

OR («YOKI» – mantiqiy qo'shish amali),

NOT («EMAS» – mantiqiy inkor amali).

Yodingizda bo'lsa, nom va qiymati o'zgaradigan miqdorlar haqida aytib o'tgan edik. Yana dasturlash tillarida quyidagilar qo'llaniladi:

Konstantalar (o'zgarmaslar) – dastur ishlashi davomida qiymati o'zgar olmaydigan miqdorlar;

O'zgaruvchilar – dastur ishlashi davomida qiymati o'zgaradigan miqdorlar;

Algebraik ifodalar – arifmetik amallar bilan bog'langan o'zgarmaslar, o'zgaruvchilar va funksiyalar;

Operatorlar – dasturlash tilining biror tugallangan amalini berish uchun mo'ljallangan buyrug'i, operatorlar BASIC da «:» bilan, PASCAL va DELPHI da «;» bilan ajratiladi;

Funksiya va protseduralar – o'z nomiga ega bo'lgan alohida dastur qismlari (bloklari). Ularga asosiy dasturdan murojaat etiladi;

Nishonlar – dasturda boshqarish uzatilayotgan operatori ko'rsatadi. Har bir dasturlash tili yuqoridagi tushunchalar bilan bog'liq o'z sintaksisiga, maxsus xizmatchi so'zlariga ega. Dastur yozishdan avval unda ishtirok etadigan miqdorlarni aniqlab olish, o'zgaruvchilarga nom berish va ularni tavsiflash (turini ko'rsatish) kerak bo'ladi. Shundan so'nggina dasturning asosiy qismi boshlanadi.

Har qanday dasturlash tili, odatda, quyidagi ikki qismdan tashkil topadi:

- tavsiflash qismi, PASCAL va DELPHI da **VAR** xizmatchi so'zi bilan boshlanib va **BEGIN** xizmatchi so'zidan oldin tugaydi;
- asosiy qism, PASCAL va DELPHI da **BEGIN** xizmatchi so'zi bilan boshladi hamda **END** xizmatchi so'zi bilan tugaydi.

Dasturlash tillarida asosan uch xil: o'zgarmas, o'zgaruvchi (masalan, A tokcha) va massiv (jadval, masalan, biz ishlatgan qavatli tokchalar) ko'rinishidagi miqdorlar qo'llaniladi. Ular belgili, satrli, mantiqiy va sonli turdagi qiymatlarni qabul qilishi mumkin.

O'zgarmas miqdorlar

Belgili o'zgarmaslar ajratish belgisi ichiga olingan bitta belgi - harf, raqam yoki maxsus belgidan iborat. Masalan:

BASIC	PASCAL	DELPHI
«a»; «B»; «9»; «-» va hokazo	'a'; 'B'; '9'; '-' va hokazo	'a'; 'B'; '9'; '-' va hokazo

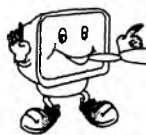
Satrli o'zgarmaslar uzunligi 255 ta belgidan oshmagan va apostrof ichiga olingan harf, raqam va maxsus belgilar ketma-ketligidan iborat. Masalan:

BASIC	PASCAL	DELPHI
«Toshkent»; «A 549»; «***.»; «37%»; «A = »; »...-« va hokazo	'Toshkent'; 'A 549'; '***.'; '37%'; 'A = '; '...-' va hokazo	'Toshkent'; 'A 549'; '***.'; '37%'; 'A = '; '...-' va hokazo

Mantiqiy o'zgarmaslar faqat True (rost) yoki False (yolg'on) qiymatlardan birini qabul qiladi.

Sonli o'zgarmaslar ikki turda – **butun** yoki **haqiqiy** bo'lishi mumkin. Haqiqiy sonlar o'z navbatida qo'zg'almas nuqtali va qo'zg'aluvchi nuqtali sonlarga bo'linadi.

Qo'zg'almas nuqtali sonlar – o'nli kasr ko'rinishidagi sonlardir. Masalan: – 2.753; 283.45; 0.517; – 0.0013.



Dasturlash tilida o'nli kasrlarning butun va kasr qismini ajratuvchi «vergul» o'rniga «nuqta» yoziladi

Qo'zg'aluvchi nuqtali sonlar – eksponensial ko'rinishda ifodalangan sonlardir. Sonlarni bu usulda yozish juda kichik yoki juda katta sonlarni ifodalashda qo'l keladi. Masalan, $3400000000 = 3,4 \cdot 10^9$ soni $3.4E9$ kabi eksponensial ko'rinishda yoziladi. **E** harfidan oldin yozilgan son mantissa, **E** harfidan keyin yozilgan son esa tartib deb ataladi. Mantissa butun yoki qo'zg'aluvchi nuqtali shaklda berilishi, tartib esa faqat butun son bo'lishi mumkin.

10.1-misol

$$37.3879 E-3 = 0.0373879; \quad 5.31 E+5 = 531000; \\ -0.075 E-5 = -0.00000075; \quad -2.37 E-4 = -0.000237$$

O'zgaruvchi miqdorlar

Dasturning bajarilish jarayonida qiymati o'zgaradigan miqdorlar **o'zgaruvchi miqdorlar** yoki qisqacha **o'zgaruvchilar** deyiladi. O'zgaruvchilar barchasida kamida 1 ta belgi, BASIC da 40 ta belgidan oshmaydigan, PASCAL da 63 ta belgidan oshmaydigan, DELPHI da (nom va keyingi yozuvlar sig'ishiga qarab) 1000 ta belgidan ortiq o'z nomiga ega bo'ladi. O'zgaruvchi nomi uchallasida lotin harfi yoki PASCAL va DELPHI da tagchiziq belgisidan boshlanadi. O'zgaruvchi nomida faqatgina raqamlar, lotin harflari va aytib o'tilgan holda, tagchiziq belgisi qo'llanilishi mumkin. Masalan:

abc; son; mening_ismim; sinf_9.

Bu tillarda o'zgaruvchilarning nomida qo'llanilgan katta va kichik lotin harflari farqlanmaydi. Masalan, *karra*, *Karra*, *kArRa* nomlar bitta o'zgaruvchini bildiradi. Bunday yozish xizmatchi so'zlar uchun ham o'rinlidir.

PASCAL va DELPHI: o'zgaruvchilar dasturning tavsiflash qismida albatta tavsiflanishi, ya'ni ularning turi ko'rsatilgan bo'lishi lozim. Dasturda o'zgaruvchilarni tavsiflash **Var** xizmatchi so'zi bilan boshlanadi.

BASIC, PASCAL va DELPHI: butun sonli qiymatlar qabul qiladigan o'zgaruvchilar **butun sonli o'zgaruvchilar** deyiladi. PASCAL va DELPHI da o'xshash 5 tur o'zgaruvchi bo'lib, bir-

biridan qabul qiladigan qiymatlarining chegarasi (diapazoni) va kompyuter xotirasidan egallaydigan joyi (hajmi) bilan farqlanadi. Quyidagi jadvallarda butun sonli o'zgaruvchilarni tavsiflash uchun maxsus so'zlar, ularga mos qiymatlar chegarasi va egallaydigan xotira hajmi keltirilgan:

PASCAL da	Qiymatlar chegarasi	Egallaydigan xotira hajmi
ShortInt	-128 ... 127	8 bit
Integer	-32768 ... 32767	16 bit
LongInt	-2147483648 .. 2147483647	32 bit
Byte	0... 255	8 bit
Word	0... 65535	16 bit

DELPHI da	Qiymatlar chegarasi	Egallaydigan xotira hajmi
ShortInt	-128 ... 127	8 bit
SmallInt	-32768 ... 32767	16 bit
Integer	-2147483648 ... 2147483647	16 bit
LongInt	-2147483648 ... 2147483647	32 bit
Int64	-9223372036854775808 ... 9223372036854775807	64 bit
Byte	0... 255	8 bit
Word	0... 65535	16 bit
Longword	0...4294967295	32 bit

10.2-misol

Var

N, k : Integer;

tartib_raqam : Byte;

PASCAL va DELPHI: haqiqiy sonli qiymatlar qabul qiladigan o'zgaruvchilar **haqiqiy sonli o'zgaruvchilar** deyiladi. Ularning turlar quyidagi jadvalda keltirilgan:

PASCAL da	Qiymatlar chegarasi	Razryadi	Egallaydigan xotira hajmi
Real	$-2,9 \cdot 10^{39} \dots 1,7 \cdot 10^{38}$	11-12	6 bayt
Single	$-1,5 \cdot 10^{45} \dots 3,4 \cdot 10^{38}$	7-8	4 bayt
Double	$-5,0 \cdot 10^{324} \dots 1,7 \cdot 10^{308}$	15-16	8 bayt
Extended	$-3,4 \cdot 10^{4932} \dots 1,1 \cdot 10^{4932}$	19-20	10bayt
Comp	$-9,2 \cdot 10^{18} \dots 9,2 \cdot 10^{18}$	19-20	8 bayt

DELPHI da	Qiymatlar chegarasi	Razryadi	Egallaydigan xotira hajmi
Real48	$-2,9 \cdot 10^{39} \dots 1,7 \cdot 10^{38}$	11-12	6 bayt
Single	$-1,5 \cdot 10^{45} \dots 3,4 \cdot 10^{38}$	7-8	4 bayt
Real	$-5,0 \cdot 10^{324} \dots 1,7 \cdot 10^{308}$	15-16	8 bayt
Double	$-5,0 \cdot 10^{324} \dots 1,7 \cdot 10^{308}$	15-16	8 bayt
Extended	$-3,6 \cdot 10^{4951} \dots 1,1 \cdot 10^{4932}$	19-20	10bayt
Comp	$-9,2 \cdot 10^{18} \dots 9,2 \cdot 10^{18}$	19-20	8 bayt
Currency	$-922337203685477.5808 \dots$ 922337203685477.5807	19-20	8 bayt

Jadvaldagi «Razryadi» sonning aniq raqamlari sonini bildiradi. Juda ko'p hollarda real turidagi o'zgaruvchilardan foydalanish yetarli bo'ladi.

PASCAL va DELPHI: satrli o'zgaruvchilarni tavsiflash uchun **String** maxsus so'zi qo'llaniladi. Bunday o'zgaruvchilar uchun PASCALda kompyuter xotirasidan 256 bayt (256 ta belgi uchun) joy ajratiladi. DELPHI da esa bu ko'rsatkich 2 Gbayt gacha bo'lishi mumkin. Agar satrli o'zgaruvchi qabul qiladigan satrdagi belgilar soni dastur ishlashi davomida ma'lum miqdordan, masalan, 10 ta belgidan oshmasa, kompyuter xotirasini tejash maqsadida, uni String[10] orqali tavsiflash maqsadga muvofiq.

10.3-misol

var

qator : String; {qator nomli o'zgaruvchiga xotiradan 256 bayt ajratildi}

_satr : String[24]; {_satr nomli o'zgaruvchiga xotiradan 24 bayt ajratildi}

PASCAL va DELPHI: mantiqiy o'zgaruvchilar Paskalning **Boolean** maxsus so'zi orqali tavsiflanadi.

10.4-misol

var

natija : Boolean;

katta, kichik : Boolean;

BASIC: o'zgaruvchilar tavsiflanishi shart emas, agar tavsiflash zarurati bo'lsa o'zgaruvchini qo'llashdan avval tavsiflanadi, masalan:

DIM N AS INTEGER

DIM B AS DOUBLE

Butun sonli o'zgaruvchilar tavsiflanmasdan **nom%** kabi ham qo'llanilaveradi. Chegaralashlar quyidagicha:

Turi	Qiymatlar chegarasi
Butun	-32768 ... 32767
Uzun butun	-2147483648... 2147483647
Oddiy musbat haqiqiy	$2.802597 \cdot 10^{-45} \dots 3.402823 \cdot 10^{38}$
Oddiy manfiy haqiqiy	$-2.802597 \cdot 10^{-45} \dots -3.402823 \cdot 10^{38}$
Ikkiilangan musbat haqiqiy	$4.940656458412465 \cdot 10^{-324} \dots$ $1.79769313486231 \cdot 10^{308}$
Ikkiilangan manfiy haqiqiy	$-4.940656458412465 \cdot 10^{-324} \dots$ $-1.79769313486231 \cdot 10^{308}$

BASIC: belgisi – satrli o'zgaruvchilarning oxirida \$ yoziladi, ular uchun kompyuter xotirasidan 0 baytdan 32767 baytgacha joy ajratiladi; **mantiqiy o'zgaruvchilarni** tavsiflash shart emas.

Bu yerda bob boshida aytib o'tilgan tokchani xususiyati o'zgaruvchining turi bilan mos tushmoqda. Agar o'zgaruvchi qiymati quyi chegaradan kichik bo'lsa yoki yuqori chegaradan katta bo'lsa, xato natijaga olib keladi yoki **INKOR** holat yuzaga keladi.

Jadval ko'rinishidagi miqdorlar

Kundalik hayotimizda ko'p turdagi jadvallardan foydalanamiz: dars jadvali, shaxmat yoki futbol o'yinlari bo'yicha musobaqa jadvali, lotereya jadvali, karra jadvali, matematik jadval va boshqalar. Biz foydalangan tokchalar ham jadvalga misol bo'ladi. Jadvalni tashkil etuvchilar uning **elementlari** deyiladi.

Jadval ko'rinishidagi miqdorlar bir o'lchovli (chiziqli), ikki o'lchovli (to'g'ri to'rtburchakli), uch o'lchovli va hokazo bo'ladi. Biz, odatda, chiziqli va to'g'ri to'rtburchakli jadvallardan foydalanamiz.

Chiziqli jadvallar satr yoki ustun shaklida ifodalanadi. Masalan, sinfingizdagi o'quvchilar ro'yxati sinf jurnalida ustun shaklidagi jadval ko'rinishida yozilgan. O'quvchilarning familiyalari bu jadvalning elementlarini tashkil etadi. Ularning har biri o'z tartib raqamiga ega va har bir tartib raqamga faqat bitta o'quvchining familiyasi mos keladi.

Ikki o'lchovli jadvallar ustunlar va satrlardan tashkil topadi (elektron jadvallarga oid mavzularni eslang). Ularning elementlari

ustun va satrlar kesishgan kataklarda joylashadi. Bunday jadvallarda biror elementni ko'rsatish uchun uning nechanchi satr va nechanchi ustunda joylashganligini, ya'ni satr va ustun bo'yicha tartib raqamlarini bilish kerak bo'ladi. Demak, ikki o'Ichovli jadvalning har bir elementiga ikkita tartib raqami (sitr va ustun bo'yicha) mos keladi.

10.5-misol

Guruhlardagi a'lochi o'quvchilar sonini ifodalovchi jadval tuzing.

semestr guruh	I	II
1011	5	6
1012	4	5
1013	5	4

Dasturlash tillarida jadvallar bilan ishlash uchun **massiv** tushunchasi kiritilgan. **Massiv** – jadval ko'rinishidagi miqdor bo'lib, u ma'lum (aniq) sondagi bir turli va tartiblangan (tartib raqamiga ega) elementlar majmuidan iborat. Massiv elementlarining tartib raqami butun sonlarda ifodalanadi. Ular **BASIC** da **manfiy emas**, **PASCAL** va **DELPHI** da **manfiy** bo'lishi ham mumkin.

Dasturlash tillarida har bir massiv o'z nomiga ega bo'lib, ularni nomlash o'zgaruvchilarni nomlash kabidir. Masalan:

a5, dars_jadvali, tub_sonlar.

Massiv element (tokcha) larning tartib raqami indeks deb ataladi. Indeks **BASIC** da oddiy qavs, **PASCAL** va **DELPHI** da kvadrat qavs ichida ko'rsatiladi. Masalan, **PASCAL** va **DELPHI** da a[5] yozuvi – a nomli massivning beshinchi elementini bildiradi.

10.6-misol

A nomli 7 ta elementdan iborat chiziqli jadvalni tasvirlang.

Tartib raqami	1	2	3	4	5	6	7
Qiymati	3 A[1]	-7 A[2]	4 A[3]	1 A[4]	-1 A[5]	0 A[6]	5 A[7]

Ikki o'Ichovli massiv elementlari ikkita indeks orqali aniqlanib, ular o'zaro vergul bilan ajratib yoziladi va birinchi indeks satr

tartib raqamini, ikkinchi indeks ustun tartib raqamini bildiradi. Masalan, *jadval*[12,8] yozuvi – jadval nomli massivning 12-satri va 8-ustuni kesishgan katakda joylashgan elementini bildiradi.

10.7-misol

S nomli 4x5 elementdan iborat to'g'ri to'rtburchakli jadvalni tasvirlang.

		Ustun bo'yicha tartib raqami				
		1	2	3	4	5
Satr bo'yicha tartib raqami	1	3.2 S[1,1]	1.37 S[1,2]	-1.25 S[1,3]	7.12 S[1,4]	-11.4 S[1,5]
	2	0,5 S[2,1]	1.1 S[2,2]	1.2 S[2,3]	-1,1 S[2,4]	4.22 S[2,5]
	3	-0.1 S[3,1]	1.01 S[3,2]	71.2 S[3,3]	4.1 S[3,4]	-4.11 S[3,5]
	4	6.3 S[4,1]	-7.01 S[4,2]	1.5 S[4,3]	7.5 S[4,4]	-1.09 S[4,5]

Dasturlash tillarida massivlar o'zgaruvchilar kabi tavsiflanishi zarur. Buning uchun **BASIC** ning **DIM**, **PASCAL** va **DELPHI** ning **Array** – xizmatchi so'zi qo'llaniladi. **BASIC** da **DIM** so'zidan keyin massiv nomi va qavs ichida birinchi hamda oxirgi elementlarning tartib raqamlari yoki oxirgi elementning tartib raqami (bu holda hisob 0 dan boshlanadi) yoziladi. Masalan:

DIM A(1 TO 100) yoki **Dim** A(5) yoki **DIM** B(N).

Array so'zidan keyin **PASCAL** va **DELPHI** da kvadrat qavs ichida massivning birinchi hamda oxirgi elementlarining tartib raqamlari o'zaro ikkita nuqta bilan ajratib yoziladi. Davomida *Of* – xizmatchi so'zi, undan keyin massiv elementlarining turi yoziladi. Masalan:

var

alifbo: array[1..29] of Char; {1 dan 29 gacha tartib raqamli elementlari belgili miqdorli bo'lgan alifbo nomli massiv}

b5: Array[-2..100] of integer; {b5: -2 dan 100 gacha tartib raqamli butun sonli massiv}

bma: array[1..10,1..20] of string; {bma - satrli miqdorlardan iborat ikki o'lchamli massiv}



Demak, massiv (jadval ko'rinishidagi miqdor) deganda, yagona nom bilan belgilangan, bir turdagi, tartiblangan miqdorlarning majmui tushuniladi.

10.8-misol

Bir o'lchovli A jadval beshta elementga ega bo'lsin:

Tartib raqami	1	2	3	4	5
Qiymati	3	2	12	10	-8

PASCAL va **DELPHI** da bu jadval elementlari quyidagicha tasvirlanadi:

$a[1] := 3; a[2] := 2; a[3] := 12; a[4] := 10; a[5] := -8;$

Massiv elementlari indeksini biror butun qiymatli o'zgaruvchi (masalan, i) orqali ifodalash mumkin. Agar $i = 3$ bo'lsa, $a[i] = 12$, $i = 5$ bo'lsa, $a[i] = -8$ bo'ladi va hokazo.

10.9-misol

Ikki o'lchovli butun sonli qiymatlar qabul qiluvchi B massiv berilgan bo'lsin:

$$B = \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \end{bmatrix} = \begin{bmatrix} 3 & 10 & 5 \\ 2 & 7 & 9 \end{bmatrix} = \begin{bmatrix} B_{ij} \end{bmatrix}$$

bu yerda $i = 0, 1$ va $j = 0, 1, 2$ (i – satr tartib raqami, j – ustun tartib raqami). U Paskalda quyidagicha tavsiflanadi:

var b: array[0..1, 0..2] of Integer;

Massivning elementlari $B[0,0]$, $B[0,1]$, $B[0,2]$, $B[1,0]$, ... kabi yoziladi. Umumiy holda indeks sifatida o'zgaruvchi yoki ifoda qo'llaniladi. Masalan, $I = 0$, $J = 2$ bo'lsa, $B[I,J] = 5$, $B[I+1, J-2] = 2$ bo'ladi.

Biz jadvallarning faqat chiziqli va to'g'ri to'rtburchakli shakllari bilan tanishdik. Aslida **PASCAL** va **DELPHI** tilida ko'p o'lchovli (255 tagacha) jadval ko'rinishidagi miqdorlardan ham foydalanish mumkin. Bunday jadvallarni tavsiflashga bir necha misollar keltiramiz.

1) *var s: array[1..4, 1..7, 0..10] of Byte; {s - Byte turli 3 o'lchovli jadval}*

2) *var t, k: array[1..100, 1..80, 1..50] of string; {t va k - 3 o'lchovli satrli jadvallar}*

3) *var f: array[-5..10, 0..10, 2..10] of char; {f - 3 o'lchovli belgili jadval}*

Shuni ta'kidlash joizki, dasturlash tillarida juda ko'p masalalarni hal etishda massivlardan foydalanish dasturchiga katta imkoniyatlar ochib beradi.

Ba'zi standart funksiyalar va algebraik ifodalar

Funksiya tushunchasi sizga matematika fanidan ma'lum. Funksiyalarning xususiyatlariga qarab turli sinflarga ajratgansiz. Masalan, chiziqli, kvadratik, trigonometrik va hokazo. Shunday funksiyalarning ba'zilaridan dasturlash tillarida ham foydalaniladi. Dastur «tushunadigan» funksiyalar **standart funksiyalar** deb yuritiladi.

Quyida ba'zi standart funksiyalarni keltiramiz:

PASCAL va DELPHI	Izoh	BASIC
abs(x)	«x» ning absolut qiymati (moduli) - $ x $	abs(x)
sin(x)	«x» ning sinusi (radian o'lb.) - $\sin x$	sin(x)
cos(x)	«x» ning kosinusi (radian o'lb.) - $\cos x$	cos(x)
sqr(x)	«x» ning kvadrat ildizi - \sqrt{x}	sqr(x)
sqr(x)	«x» ning kvadrati - x^2	x^2
exp(x)	e^x ($e = 2.718282... $)	exp(x)
round(x)	«x» ning yaxlitlangan butun qismi $[x]$	int(x)
trunc(x)	«x» ning yaxlitlanmagan butun qismi	fix(x)

Dasturlash tilida **algebraik ifodalar** arifmetik amallar bilan bog'langan o'zgarmas va o'zgaruvchi miqdorlar, funksiyalar yordamida tashkil topadi va bir satrda yoziladi. Satrdan pastga tushirib yoki yuqoriga ko'tarib yozish mumkin emas. Masalan,

$3ab^2$ ifoda **3*a*sqr(b)** kabi yozilsa, $\frac{a}{b^2}$ ifoda **a/sqr(b)** kabi yoziladi.

Ifodalarni yozishda amallarni bajarish tartibini ko'rsatish uchun faqat kichik qavslar qo'llaniladi. Qavs ichidagi amallarni bajarish chapdan o'ngga qarab, matematikada qabul qilingan odatdagi tartib saqlangan holda ketma-ket amalga oshiriladi:

- funksiya qiymatlari hisoblanadi;
- ko'raytirish va bo'lish amallari bajariladi;
- qo'shish va ayirish amallari bajariladi.

Masalan, $\frac{a+b}{c}$ arifmetik ifodani yozilishi **(a+b)/c** kabi bo'lib, uni hisoblashda dastlab qavs ichidagi amal, ya'ni **a+b** bajariladi, so'ngra natija **c** ga bo'linadi.

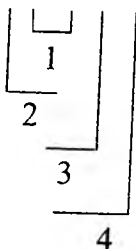
10.10-misol

R va H o'zgaruvchilarning ma'lum qiymatlarida quyidagi ifodaning qiymati hisoblansin:

$$\frac{1}{3}\pi R^2 H$$

Bu ifoda $\text{Pi}*\text{sqr}(r)*h/3$ kabi yoziladi. Bunda amallar quyidagi tartibda bajariladi:

$$\text{Pi}*\text{sqr}(r)*h/3$$



- | | |
|----|-------------------------------|
| 1. | $\text{sqr}(r)$ |
| 2. | $\text{Pi}*\text{sqr}(r)$ |
| 3. | $\text{Pi}*\text{sqr}(r)*h$ |
| 4. | $\text{Pi}*\text{sqr}(r)*h/3$ |

Shuni eslatib o'tish lozimki, ikkita amal ketma-ket kelganda ifodani qavssiz yozish mumkin emas. Masalan, $\sqrt{a^2 - b^2}$ ifodani $\text{sqr}(\text{sqr}(a) - \text{sqr}(b))$ kabi, $|x + \text{tg}x|$ ifoda $\text{abs}(x + \text{tan}(x))$ ko'rinishida yoziladi.

Ba'zi hollarda dasturlash tilida yozilgan ifodani odatdagi matematik ko'rinishda yozish talab etiladi. Masalan, $0.5*(\sin(x) + \cos(x))$ dasturlash tilida yozilgan ifoda matematik ko'rinishda quyidagicha ifodalanadi:

$$\frac{1}{3}(\sin x + \cos x).$$

Daraja bilan ishlash biroz farqlanadi. **PASCAL** va **DELPHI** da a^3 ifodani $a*a*a$ yoki $\text{sqr}(a)*a$ ko'rinishida, a^4 ifodani esa $\text{sqr}(\text{sqr}(a))$ ko'rinishida yozishga to'g'ri keladi. **BASIC** da a^n ifodani $a^{\wedge}n$ ko'rinishida yozish mumkin.

Umuman, a^b ko'rinishdagi ifoda uchun matematikada $a^b = e^{b*\ln a}$ formula o'rinli. Shuning uchun **PASCAL** va **DELPHI** da $\text{exp}(b*\ln(a))$ ko'rinishda yoziladi.

10.11-misol

$\frac{x - y}{x^5 - y^3}$ algebraik ifodani dasturlash tilida yozing.

Yechish: Bu ifodani **PASCAL** va **DELPHI** da bir necha xil usulda tasvirlash mumkin. Shulardan bittasi quyidagicha:
 $(x-y)/(\exp(5*\ln(x))-sqr(y)*y)$.

O'zlashtirish va ma'lumotlarni ekranga chiqarish operatorlari

Dasturlash tillaridagi o'zlashtirish operatorlarini taqqoslaymiz:

O'zlashtirish operatorlari		
Algoritmik tilimizda	BASIC	PASCAL va DELPHI
o'tkaz <ifoda>,o'zgaruvchi	o'zgaruvchi = <ifoda>	o'zgaruvchi := <ifoda>;
Misollar		
o'tkaz 1,K	K = 1	K := 1;
o'tkaz 1+N,K	K = N+1	K := N+1;
o'tkaz a*5,b	b = a*5	b := a*5;

Endi ma'lumotlarni ekranga chiqarish operatorlarini taqqoslaymiz:

Ma'lumotlarni ekranga chiqarish operatorlari		
Algoritmik tilimizda shart emas edi	BASIC	PASCAL va DELPHI
	Print <chiqarish ro'yxati>	Write(chiqarish ro'yxati); yoki Writeln(chiqarish ro'yxati);
	Misol	
	K = 55/11; b="K=1 mantiqiy ifoda" Print 0; "-son" Print K, K= 1; Print b	K:=55/11; b='K=1 mantiqiy ifoda'; Writeln(0,'-son'); Write(K,' ',K= 1); Writeln(b);
	Ekranida	
	0-son 5 OK=1 mantiqiy ifoda	0-son 5 OK=1 mantiqiy ifoda
	Misol	
	Print 7: Print: Print 21	Writeln(7); Writeln; Writeln(21);
	Ekranida	
	7 21	7 21

BASIC: *Print* operatori ro'yxatdagi o'zgaruvchilar orasida ";," qo'yilgan bo'lsa, o'zgaruvchilar qiymati orasiga bitta belgi sig'adigan bo'sh joy qoldiradi; ";," qo'yilgan bo'lsa, o'zgaruvchilar qiymati orasiga 17 belgi sig'adigan bo'sh joy qoldirib ekranga chiqaradi. Agar ro'yxat oxirida ";," yoki ";," yozilgan bo'lsa, u holda keyingi *Print* ro'yxatidagi qiymat shu satr davomidan mos bo'sh joy qoldirib chiqariladi. Agar ro'yxatsiz *Print* (yoki bo'sh *Print* ham deyishadi) yozilgan bo'lsa, u holda bo'sh satr qoldirish uchun yoki agar avvalgi *Print* ro'yxati oxirida ";," yoki ";," yozilgan bo'lsa, satrni to'lgan hisoblanishi uchun qo'llaniladi. *Print* o'rniga "?" belgisini yozish ham mumkin.

PASCAL va DELPHI: *Write* va *Writeln* operatorlarining farqi shundaki, *Write* operatori yordamida ma'lumotlar ekranga chiqarilgach yurgich ekranning shu satrida qoladi, ya'ni ekranga keyingi chiqariladigan ma'lumotlar shu satrga (yurgich turgan joydan) chiqariladi. *Writeln* operatorida esa ma'lumotlar ekranga chiqarilgach yurgich keyingi satr boshiga o'tadi.

Ma'lumotlarni xotiraga muloqot usulida kiritish operatori

Ma'lumotlarni ekranga chiqarish operatorlari		
	BASIC	PASCAL va DELPHI
Algoritmik tilimizda shart emas edi	Input < kiritish ro'yxati > yoki Input "izoh" < kiritish ro'yxati >	Read(kiritish ro'yxati); yoki Readln(kiritish ro'yxati);
	Misol	
	Input a,b,d Input "Kiriting a,b,d: ",a,b,d Input "Kiriting a,b,d: ";a,b,d	Readln(a,b,d); Writeln('Kiriting: a,b,d: '); Read(a,b,d); Write('Kiriting: a,b,d: '); Read(a,b,d);
	Ekranda	
	1,2,3 Kiriting a,b,d: 1,2,3 Kiriting a,b,d: ? 1,2,3	123 Kiriting a,b,d: 1,2,3 Kiriting a,b,d: 1,2,3
	Misol	
	Input a Input "b="";b Input "d=""; d	Readln(a); Write('b='); Readln(b); Write('d='); Read(d);
	Ekranda	
	1 b=2 d=3	1 b=2 d=3

Kiritish operatori dastur ishlashini to'xtatadi va ro'yxatdagi o'zgaruvchilarga klaviatura orqali qiymat berilishini kutadi. Agar ro'yxatda bir nechta o'zgaruvchi bo'lsa, ularning qiymatlari kiritib bo'lingach <ENTER> klavishi bosiladi.

BASIC: Agar ro'yxatda bir nechta o'zgaruvchi bo'lsa, ularning qiymatlari o'zaro ", " bilan ajratib kiritiladi. **Input** operatorining izohli holda kiritilayotgan qiymatlarga mos izoh berish mumkin bo'ladi.

PASCAL va **DELPHI:** Agar ro'yxatda bir nechta o'zgaruvchi bo'lsa, ularning qiymatlari o'zaro probel (bo'sh joy) bilan ajratib kiritiladi. **Read** va **Readln** operatorlarining farqi shundaki, **Read** operatori yordamida ma'lumotlar kiritilgach yurgich ekranning shu satrida qoladi, ya'ni kiritiladigan ma'lumotlar shu satrda (yurgich turgan joyda) kiritiladi. **Readln** operatorida esa ma'lumotlar kiritilib bo'lingach, yurgich keyingi satr boshiga o'tadi.

Chiziqli dasturlar tuzish

Dasturlash tillari imkoniyatlari va operatorlari haqida ko'p gapirish mumkin. Har bir dasturlash tili uchun alohida qo'llanmalar yozish mumkin.

Siz zerikib qolmasligingiz uchun dastur tuzishni o'rganishni boshlaymiz, yo'q, unday emas, yozgan algoritmlaringizning ko'rinishini kompyuter tushunadigan qilib o'zgartirishni boshlaymiz. Chiziqli algoritmlarning dastur shaklida yozilishi **chiziqli dastur** deb ataladi. Demak, chiziqli dasturdagi amallarni bajarishda hech qanday shart tekshirilmaydi, barcha operatorlar (algoritmdagi ko'rsatmalar) kelish tartibida bajariladi. Yana shuni yodda tutingki, biz **DELPHI** tilini **PASCAL** ga o'xshab ketadigan **Console Application** ilovasida dastur tuzamiz. Shuni inobatga olgan holda dasturning boshida **{\$apptype console}** direktivasi yozilgan bo'lishi kerak.

10.1-masala

Berilgan a va b sonlarda $S = a + b \cdot a - a \cdot (21 - b) : 7$ ifodaning qiymatini hisoblang.

Javob. **PASCAL** va **DELPHI** da o'zgaruvchilarni tavsiflash shart. Kiritilayotgan qiymatlar haqida hech nima aytilmagan bo'lsa, ularni, odatda, haqiqiy, ya'ni **REAL** turli deb olinadi. O'zimizga oson bo'lishi uchun oraliq o'zgaruvchilar d va g ni kiritib dasturlarni taqqoslash jadvalini yozamiz:

Algoritmik tilimizda	BASIC	PASCAL va DELPHI
o'tkaz 21-7, d o'tkaz b-a, g o'tkaz a-d, d o'tkaz d/7, d o'tkaz a+g-d, S	INPUT a INPUT b d=21-b g=b*a d=a*d d=d/7 S=a+g-d Print S	var s,a,b: real; g,d: real; begin Read(a); Read (b); d:=21-b; g:=b*a; d:=a*d; d:=d/7; S:=a+g-d; Write(S); end
Dasturni ishga tushirish	F5	PASCAL: Ctrl+F9; DELPHI: F9

10.2-masala

Radiusi butun R bo'lgan aylananing uzunligini hisoblash dasturi tuzilsin.

Yechim: Aylananing uzunligini hisoblash formulasini esga olamiz: $L=2\pi R$. Dasturlash tilida esa o'ng tomondagi ifoda $2*3.14*R$ ko'rinishga yoziladi. Dasturda ikkita o'zgaruvchi qatnashadi: R va L . Masala shartiga ko'ra butun son. Shu sababli R o'zgaruvchi turini **Integer** deb olamiz. Aylananing uzunligi L esa ko'paytmada π qatnashgani uchun, albatta haqiqiy (**Real**) turli bo'ladi. Aytilganlarni hisobga olib quyidagi algoritm va dasturlarni tuzamiz:

Algoritmik tilimizda	BASIC	PASCAL va DELPHI
o'tkaz 2*3.14,L o'tkaz L-R,L	INPUT "R ni kiriting ", R L = 2*3.14*R Print "L="; L ; "birlik"	Var R: Integer; L: Real; Begin Write('R ni kiriting '); ReadLn(R); L := 2*3.14*R; WriteLn('L=',L,'birlik.');
Dasturni ishga tushirish	F5	PASCAL: Ctrl+F9; DELPHI: F9

11.3-masala

Tomonlari mos ravishda, juft natural a, b, c son bo'lgan ixtiyoriy uchburchakning yuzini kvadratini Geron formulasi orqali hisoblash dasturini tuzing.

Yechim. Uchburchak yuzining kvadratini hisoblash uchun yarim perimetr va Geron formulasini yozib olamiz:

$$p = \frac{a+b+c}{2},$$

$$G^2 = \left(\sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)} \right)^2 = p \cdot (p-a) \cdot (p-b) \cdot (p-c).$$

Dastur tuzishda BASIC da muammo yo'qku, lekin PASCAL va DELPHI da yana o'zgaruvchilarning turini aniqlab olishimiz shart. Berilgan a, b, c sonlar natural (demak, butun) hamda juft. U holda P ham juft va 2 ga bo'lganda butun son hosil bo'ladi. Geron formulasida faqat ayirma va ko'paytmalar bor, demak, uni qiymatining kvadrati ham butun. Lekin, PASCAL va DELPHI tillari bo'lish natijasini doim **haqiqiy** deb hisoblaydi. Shuning uchun, P ham G ham haqiqiy (REAL) turdagi o'zgaruvchi deb tavsiflanadi.

Algoritmik tilimizda	BASIC	PASCAL va DELPHI
o'tkaz a+b, p o'tkaz c+p, p o'tkaz p/2, p o'tkaz p-a, a o'tkaz p-b, b o'tkaz p-c, c o'tkaz p-a, G o'tkaz G-b, G o'tkaz G-c, G	INPUT "a ni kiriting ", a INPUT "b ni kiriting ", b INPUT "c ni kiriting ", c P=(a+b+c)/2 Gkv= p*(p-a)*(p-b)*(p-c) Print "G kvadrat="; Gkv	Var a,b,c: Integer; P, Gkv: Real; Begin Write('a ni kiriting '); ReadLn(a); Write('b ni kiriting '); ReadLn(b); Write('c ni kiriting '); ReadLn(c); P:=(a+b+c)/2; Gkv:=p*(p-a)*(p-b)*(p-c); WriteLn('G kvadrat=',Gkv); End.

Dastur tuzish oson tuyilayotgan bo'lsa, quyidagi masalani hal eting.

10.4-masala

A va B sonlar berilgan. Qo'shimcha o'zgaruvchi kiritmasdan A va B sonlarning qiymatini almashtiring ya'ni, masalan, A = 7 va B = 21 bo'lsa, dastur ishlaganidan keyin A = 21 va B = 7 bo'lishi kerak.

O'tish va tarmoqlanish operatorlari

Biz hozirgacha chiziqli, ya'ni buyruqlari ketma-ket bajariladigan dasturlar bilan tanishdik. Lekin ko'pincha berilgan masalani hal qilishda operatorlarning bajarilish tartibini buzishga, ya'ni

boshqarishni dastur bo'yicha orqaga yoki oldinga o'tkazish zarur bo'ladi. Buning uchun dasturning boshqarish uzatilayotgan operatoriga **nishon** qo'yiladi. Nishon o'zgaruvchining nomi kabi lotin harflari va raqamlar yordamida hosil qilinadi. Masalan, 7, N1, nishon2 va hokazo.

Dasturda qo'llaniladigan nishonlar PASCAL va DELPHI tilida (BASIC da emas) dasturning tavsif qismida **Label** xizmatchi so'zi yordamida ko'rsatilishi shart.

Nishonlardan dasturda **o'tish operatori** qo'llanilsagina foydalaniladi. O'tish operatori quyidagi ko'rinishga ega:

GOTO <nishon>

Bu **GOTO** (ing. – "o'tilsin") operatori boshqarishni dasturning **nishon** qo'yilgan operatoriga uzatadi.

10.12-misol

Algoritmik tilimizda shart emas edi	BASIC	PASCAL va DELPHI
	a=15 b=13 c=a+b GoToN1 c=a-b N1 Print c	Label N1; Var a,b,c:Integer; Begin a:=15; b:=13; c:=a+b; GoToN1; c:=a-b; N1: WriteLn(c); End.

Bu dasturlarning ishlashi natijasida ekranda hosil bo'lgan c ning qiymati 28 ga tengdir. Chunki, boshqarish N1 nishonli chiqarish operatoriga uzatilgani uchun BASIC da $c = a - b$ amalni, PASCAL va DELPHI da $c := a - b$ amalni bajarmasdan o'tkazib yuboriladi.

O'tish operatorida hech qanday shart tekshirilmasdan boshqarish ko'rsatilgan nishonli operatorga uzatiladi. Lekin aksariyat masalalarni hal etishda biror shartning bajarilishiga qarab u yoki bu amallar ketma-ketligini bajarish kerak bo'ladi. Masalan, Kamaytiruvchi **musbat** yoki **juft** shartini, Robot biror yo'nalishni **bo'sh** ekanligi shartini tekshirib, xulosa asosida u yoki bu ko'rsatmalarni bajarishgan edi. Shu kabi masalalarni hal qilish uchun algoritmik tilimizdagi shartli tuzilmaga mos dasturlash tillarida **tarmoqlanish operatori** qo'llaniladi. Bu operatorning umumiy shakli quyidagicha:

BASIC:

If <shart> **Then** <operator yoki operatorlar> **Else** <operator yoki operatorlar>

PASCAL va DELPHI:

If <shart> **Then** <operator yoki operatorlar> **Else** <operator yoki operatorlar>;

Bu yerda **if**, **then** va **else** dasturlash tilining xizmatchi soʻzlari boʻlib, quyida ularning oʻqilishi va maʼnosi keltirilgan:

- **If** (if) – "agar", **Then** (zen) – "u holda", **Else** (elz) – "aks holda";
- <shart> – rost yoki yolgʻon qiymatlardan birini qabul qiluvchi mantiqiy ifoda;
- <operator yoki operatorlar> – dasturlash tilining ixtiyoriy operatori yoki operatorlari ketma-ketligi boʻlishi mumkin.

Bu tarmoqlanish operatori qoʻllanmaning V bobida keltirilgan shartli tuzilmaning ikkinchi koʻrinishining xuddi oʻzi. Yodingizda boʻlsa shartli tuzilmaning birinchi koʻrinishi ham bor edi. Dasturlash tillarida ham bor boʻlib, tarmoqlanish operatorining qisqa koʻrinishi deb ataladi:

BASIC:

If <shart> **Then** <operator yoki operatorlar>

PASCAL va DELPHI:

If <shart> **Then** <operator yoki operatorlar>;

Bu operatorlarda ham birikkan shartlardan foydalanish mumkin.

Faqat baʼzi qoidalarga rioya qilinishi shart.

10.13-misol

Parol toʻgʻri kiritilganini tekshiruvchi dastur tuzing.

Yechish: Parol "informatika" boʻlsin, u holda dastur quyidagicha boʻladi:

BASIC	PASCAL va DELPHI
<pre>Input "Parolni kiriting: ", parol IF parol="informatika" THEN Print("Parol to'g'ri") ELSE Print ("Parol noto'g'ri") END IF</pre>	<pre>Var parol:String; Begin Write('Parolni kiriting: '); ReadLn(parol); IF parol='informatika' THEN WriteLn('Parol to'g'ri') ELSE WriteLn('Parol noto'g'ri'); End.</pre>

Sintaksis qoidalari:

- **BASIC:** agar IF - THEN - ELSE bir satrga yozilsa END IF ning keragi boʻlmaydi, lekin bu holda operatorlar guruhi yozilsa baʼzan xatolikka olib kelishi mumkin; birikkan shartlar

orasiga VA, YOKI, EMAS so'zlarining ingliz varianti (dastur alifbosiga qarang) yoziladi.

- PASCAL va DELPHI: agar THEN yoki ELSE so'zlaridan keyin operatorlar guruhi yozilsa, u holda bu guruh BEGIN va END; ichida yoziladi; agar ELSE qism yozilsa, u holda undan oldingi nuqtali vergul yozilmaydi; birikkan shartlarda har bir shart alohida qavsga olinadi va ular orasiga VA, YOKI, EMAS so'zlarining ingliz varianti (dastur alifbosiga qarang) yoziladi.

10.14-misol

Uchta sondan kattasini topish (UKT) dasturini tuzing.

Yechish. Bu masalani hal etishda birikkan shartlardan foydalanamiz:

Algoritmik tilimizda	
AGAR A>=B va A>=C U HOLDA o'tkaz A, KATTA TAMOM	{A>=B va A>=C}
AGAR B>=C va B>=A U HOLDA o'tkaz B, KATTA TAMOM	{B>=C va B>=A}
AGAR C>=A va C>=B U HOLDA o'tkaz C, KATTA TAMOM	{C>=A va C>=B}
BASIC	
Input "A= ", A Input "B= ", B Input "C= ", C If A>=B AND A>=C Then KATTA=A: GOTO javob If B>=A AND B>=C Then KATTA=B: GOTO javob KATTA=C javob: Print "Sonlardan kattasi "; KATTA; "ga teng"	
PASCAL va DELPHI	
Label javob; Var A,B,C, KATTA:Real; Begin Write('A= '); ReadLn(A); Write('B= '); ReadLn(B); Write('C= '); ReadLn(C); If (A>=B) AND (A>=C) Then begin KATTA:=A; goto javob; end; If (B>=A) AND (B>=C) Then begin KATTA:=B; goto javob; end; KATTA:=C; javob: Write('Sonlardan kattasi ',KATTA,'ga teng'); End.	

Avvalgi boblardagi masalalarni dasturlash tillariga o'tkazing, shu usulda tarmoqlanish operatorini tushunib olish oson bo'ladi.

Takrorlash operatorlari

Bu bo'limda algoritmik tilimizda ko'rib o'tgan takrorlash tuzilmalarini taqqoslash yo'li bilan dasturlash tillaridagi ifodasini izohlab beramiz.

Parametrlil takrorlash
Algoritmik tilimizda
N1 DAN N2 GACHA BAJAR <takrorlanishi lozim bo'lgan ko'rsatmalar> TAMOM
bu tuzilmada sanoq N1 dan boshlanadi va toki sanoq N2 ga yetguncha bittalab oshirib boriladi. Har qadamda BAJAR va TAMOM orasidagi takrorlanishi lozim bo'lgan ko'rsatmalar bajariladi.
BASIC
For I = N1 To N2 Step N3 <operator> Next I
bu yerda For (uchun), To (gacha) va Next I (keyingi I) xizmatchi so'zlar; I - haqiqiy turdagi ixtiyoriy o'zgaruvchi bo'lib, u N3 talab oshirib (kamaytirib) boriladi va takrorlash parametri deyiladi; N1 - takrorlash parametrining qabul qiladigan boshlang'ich qiymati; N2 - takrorlash parametrining qabul qilishi mumkin bo'lgan oxirgi qiymati; <operator> - takrorlanishi lozim bo'lgan operator yoki operatorlar ketma-ketligi bo'lib, u takrorlash tanasi deyiladi. Takrorlash parametrining boshlang'ich va oxirgi qiymatlari o'zgarmas, o'zgaruvchi yoki ifoda ko'rinishida bo'lishi mumkin. Agar $N3=1$ bo'lsa, Step 1 ni yozish shart emas, N3 manfiy ham bo'lishi mumkin bu holda I ning qiymati kamayib boradi.
PASCAL va DELPHI
For I: = N1 To N2 Do <operator>;
bu yerda For (uchun), To (gacha) va Do (bajar) xizmatchi so'zlar; I - butun turdagi ixtiyoriy o'zgaruvchi bo'lib, u bittalab oshirib boriladi va takrorlash parametri deyiladi; N1 - takrorlash parametrining qabul qiladigan boshlang'ich butun qiymati; N2 - takrorlash parametrining qabul qiladigan oxirgi butun qiymati; <operator> - takrorlanishi lozim bo'lgan operator yoki operatorlar ketma-ketligi bo'lib, u takrorlash tanasi deyiladi. Takrorlash tanasini operatorlar ketma-ketligi tashkil etgan bo'lsa, ular albatta begin ko'rsatmasi bilan boshlanib, end ; ko'rsatmasi bilan tugallanadi. Takrorlash parametrining boshlang'ich va oxirgi qiymatlari o'zgarmas, o'zgaruvchi yoki ifoda ko'rinishida bo'lishi mumkin. For operatorida takrorlash parametri katta qiymatdan kichik qiymatga qarab bittalab kamayib borishi ham mumkin. Buning uchun Do xizmatchi so'zi o'rniga Downto xizmatchi so'zi qo'llaniladi:
For I: = N1 Downto N2 Do <operator>;

Shart bo'yicha takrorlash

Algoritmik tilimizda

TOKI <shart> BAJAR **<ko'rsatmalar guruhi>** **TAMOM**

Bu tuzilma qanday ishlashini ko'rib chiqamiz. Avval **TOKI** so'zidan keyingi javobi **ROST** yoki **YOLG'ON** chiqadigan savol beriladi. Agar javob **ROST** bo'lsa, **BAJAR** va **TAMOM** so'zlari orasidagi ko'rsatmalar guruhi bajariladi. Bajarish jarayoni tugagandan so'ng yana **TOKI** so'zidan keyin yozilgan savol beriladi. Agar javob **ROST** bo'lsa, yana (**BAJAR** va **TAMOM** so'zlari orasidagi) ko'rsatmalar guruhi bajariladi. Shundan keyin uchinchi marta, to'rtinchi marta va hokazo marta savolga takrorlanadi. Bu takrorlanish jarayoni savolga javob **YOLG'ON** bo'lguncha davom etaveradi va javob **YOLG'ON** bo'lgandan keyingina to'xtaydi.

BASIC

Do While <shart> **<operator>** **Loop**

bu yerda Do, While (toki) va **Loop** xizmatchi so'zlar; <shart> - oddiy yoki murakkab mantiqiy ifoda; <operator> - takrorlash tanasini tashkil etuvchi operator yoki operatorlar ketma-ketligi. Mazkur takrorlanish operatori quyidagicha ishlaydi: avval <shart> tekshiriladi, agar uning qiymati "rost" bo'lsa, takrorlash tanasini tashkil etuvchi operatorlar ishlaydi va yana <shart> teksiriladi. Bu jarayon shart "yolg'on" qiymat qabul qilgunga qadar davom etadi.

Do Until <shart> **<operator>** **Loop**

Mazkur takrorlanish operatori quyidagicha ishlaydi: avval <shart> tekshiriladi, agar uning qiymati "yolg'on" bo'lsa, takrorlash tanasini tashkil etuvchi operatorlar ishlaydi va yana <shart> teksiriladi. Bu jarayon shart "rost" qiymat qabul qilgunga qadar davom etadi.

While <shart> **<operator>** **Wend**

bu yerda **While** (toki) va **Wend** xizmatchi so'zlar; <shart> - oddiy yoki murakkab mantiqiy ifoda; <operator> - takrorlash tanasini tashkil etuvchi operator yoki operatorlar ketma-ketligi. Mazkur takrorlanish operatori quyidagicha ishlaydi: avval <shart> tekshiriladi, agar uning qiymati "rost" bo'lsa, takrorlash tanasini tashkil etuvchi operatorlar ishlaydi, va yana <shart> teksiriladi. Bu jarayon shart "yolg'on" qiymat qabul qilgunga qadar davom etadi.

PASCAL va DELPHI

While <shart> Do <operator>;

bu yerda **While** (toki) va **Do** xizmatchi so'zlar; <shart> - oddiy yoki murakkab mantiqiy ifoda; <operator> - takrorlash tanasini tashkil etuvchi operator yoki operatorlar ketma-ketligi. Agar takrorlash tanasida operatorlar ketma-ketligi yozilsa, ular begin bilan boshlanib, **end;** bilan yakunlanadi.

Mazkur takrorlanish operatori quyidagicha ishlaydi: avval **<shart>** tekshiriladi, agar uning qiymati "rost" bo'lsa, takrorlash tanasini tashkil etuvchi operatorlar ishlaydi va yana **<shart>** teksiriladi. Bu jarayon shart "yolg'on" qiymat qabul qilgunga qadar davom etadi.

Repeat
<operator>
Until <shart>

bu yerda **Repeat** (takrorla) va **Until** xizmatchi so'zlar bo'lib, **Repeat** - takrorlash boshi, **Until** - takrorlash oxirini bildiradi; **<shart>** - mantiqiy ifoda; **<operator>** - takrorlash tanasini tashkil etuvchi operator yoki operatorlar ketma-ketligi. Agar takrorlash tanasida operatorlar ketma-ketligi yozilsa, ular begin bilan boshlanib, end; bilan yakunlanadi. Ular **<shart>** bajarilmaguncha (rost qiymat qabul qilmaguncha) takrorlanaveradi.

10.5-masala

I dan N gacha tartiblangan aylana shaklida joylashtirilgan likobchalarga bittadan olma qo'yilgan. Mirkamol sanoqni 1-likobchadan boshladi va aylana bo'ylab soat mili yo'nalishida yurib, har K-olmani yedi. Sanoqda olmasi yeyilgan likobcha ishtirok etmaydi. Agar Mirkamol olmaga to'ygandan keyin M ta olma qolgan bo'lsa, qaysi likobchalardagi olmalar yeyilganini topuvchi dastur tuzing.

Bu masalani algoritmik tilimizda, BASIC, PASCAL, DELPHI da o'zingiz hal eting.

Takrorlanish operatorini tushunib olish uchun avvalgi boblardagi masalalarni yechib ko'rishni maslahat beramiz.

Protseduralar

PASCAL va DELPHI
<p>Procedure <protsedura nomi>(<o'zgaruvchilar>:<o'zgaruvchilar turi>); Protsedura - bu kichik qism dasturi bo'lib, asosiy dasturning begin xizmatchi so'zidan oldin yoziladi. Protsedura ham dastur kabi begin xizmatchi so'zi bilan boshlanadi va end; bilan tugaydi. Protsedurada ham asosiy dastur kabi lokal (ichki) o'zgaruvchilar, o'zgarmaslar va nishonlardan foydalanish mumkin. Bunda asosiy dasturdagi (global) o'zgaruvchilarning qiymatlari o'zgarmasdan qoladi. Agar <o'zgaruvchilar> dan oldin var xizmatchi so'zi qo'yilsa, berilgan o'zgaruvchilarning qiymatlari asosiy dasturga o'zgarib qaytishi mumkin.</p>
BASIC
<p style="text-align: center;">Gosub <nishon></p> <p>BASIC da protseduralar nishon orqali o'tiladigan qism dastur ko'rinishida tashkil etiladi. Bunda asosiy dasturdagi o'zgaruvchilar qism dasturida ham foydalaniladi va qiymati o'zgarishi mumkin. Protsedura asosiy dasturdan keyin yoziladi. Shu sababli asosiy dastur oxirida END operatori yozilishi shart! Protsedura oxirida RETURN operatori yoziladi. U boshqaruvni asosiy dasturdagi GOSUB operatoridan keyingi operatorga uzatadi.</p>

10.6-masala

Berilgan M va N sonlarning EKUB ini hisoblash dasturini PASCAL da tuzing.

Yechim. Dasturni Evklid algoritmiga asoslangan protsedura yordamida tuzamiz.

```
var javob,m,n:integer;
    procedure ekub(a,b:integer);
    begin
        if a > b then ekub(a-b,b);
        if b > a then ekub(a,b-a);
        if a=b then javob:=b;
    end;
begin
    readln(m,n);
    ekub(m,n);
    writeln(javob);
end.
```

Agar bu protseduraga e'tibor bergan bo'lsangiz, protsedura o'zini o'zi chaqirmoqda, ya'ni biz tuzgan dasturimizda rekursiv protsedura ishtirok etmoqda.

Qo'llanmada berilgan ma'lumotlar yordamida dasturlashning boshlang'ich qismi bilan tanishdingiz. Murakkab masalalarini hal etish uchun bu ma'lumotlar kamlik qiladi. Lekin murakkab dasturlar ham shu qismlar asosida tuziladi.

Algoritmlar va dasturlash tillarining turlari nihoyatda ko'p bo'lganligi sababli har bir yo'nalishdagi algoritmlar yoki dasturlash tillarining imkoniyatlarini ochib berish uchun alohida qo'llanmalar kerak bo'ladi.

Dasturlashni chuqur o'rganmoqchi bo'lsangiz bu kabi qo'llanmalarni o'rganib chiqing.

Nazorat savollari va topshiriqlar

1. Dastur deb nimaga aytiladi?
2. Dasturlash tili deganda nimani tushunasiz?
3. Dasturlash tilining asoschisi deb kim tan olingan?
4. Quyi va yuqori darajadagi dasturlash tillari qanday xususiyatlari bilan o'zaro farqlanadi?
5. Bugungacha ma'lum bo'lgan eng qadimgi algoritmda qanday masala hal qilingan?
6. Dasturlash tillari elektron hisoblash mashinalarining turlariga bog'liq bo'ladimi? Javobingizni asoslang.

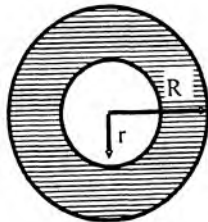
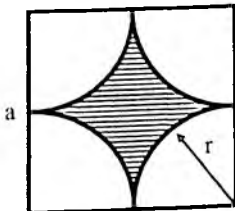
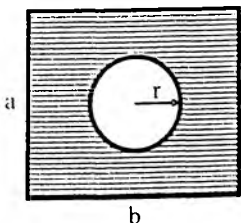
7. Yuqori darajadagi dastlabki dasturlash tilining nomi nima va u kim tomonidan yaratilgan?
8. Dasturlash tilining alifbosi haqida soʻzlab bering.
9. Operator nima?
10. Oʻzgarmas va oʻzgaruvchan miqdorlar haqida soʻzlab bering.
11. Dastur qanday qismlardan tashkil topadi?
12. Dasturning tavsiflash qismida nimalar tavsiflanadi?
13. Miqdorlar qanday turdagi qiymatlarni qabul qilishi mumkin?
14. Belgili oʻzgarmas deganda nimani tushunasiz? Misollar keltiring.
15. Satrli oʻzgarmaslarning belgili oʻzgarmaslardan farqi nimada?
16. Sonli oʻzgarmaslarning qanday turlarini bilasiz?
17. Mantiqiy oʻzgarmaslar qanday qiymatlarni qabul qilishi mumkin?
18. Oʻzgaruvchilarning oʻzgarmaslardan farqi nimada?
19. Oʻzgaruvchilarni tavsiflash uchun Paskalning qaysi xizmatchi soʻzidan foydalaniladi?
20. Sonli oʻzgaruvchilar necha turga boʻlinadi?
21. Butun sonli oʻzgaruvchilarning turlariga misollar keltiring.
22. Haqiqiy sonli oʻzgaruvchilarning turlariga misollar keltiring.
23. Belgili oʻzgaruvchilar qanday tavsiflanadi? Misollar keltiring.
24. Satrli oʻzgaruvchilar qanday tavsiflanishi mumkin? Misollar keltiring.
25. Hayotda uchraydigan jadval koʻrinishidagi miqdorlarga misollar keltiring.
26. Chiziqli massiv qanday oʻlchovlarda boʻladi?
27. Massivda indeks nima uchun zarur?
28. Massiv elementlarining indeksleri qanday qiymatlar qabul qilishi mumkin?
29. Jadval koʻrinishidagi miqdorlarning turlarini qanday ajratish mumkin?
30. Qanday funksiyalar standart funksiya deyiladi?
31. Standart funksiyalarning odatdagi va dasturlash tilida yozilishining qanday farqi bor?
32. Standart funksiyalarning qaysilarini bilasiz? Sanab bering.
33. Algebraik ifodalar nimalardan tashkil topadi?
34. Arifmetik amallarning bajarilish tartibi qanday?
35. Biror ifodada bir xil amallar qatnashsa, ularning bajarilish tartibi qanday boʻladi?
36. Amallarni bajarish tartibini oʻzgartirish uchun nimalardan foydalaniladi?
37. Input, Print, Read, Write soʻzlari qanday maʼno beradi?
38. Maʼlumotlarni muloqot holida kiritish operatorining vazifasini aytib bering.
39. Maʼlumotlarni kiritish operatorining necha xil koʻrinishi bor? Ularning farqi nimada?
40. Oʻzgaruvchilarga qiymat berishda oʻzlashtirish operatorining qulaylik tomonini tushuntiring.
41. Oʻzgaruvchilarga qiymat berishda kiritish operatorining qulaylik tomonini tushuntiring.

42. O'tish operatori nima uchun qo'llaniladi?
43. O'tish operatorining umumiy ko'rinishi qanday?
44. Dasturda nishonlar nima uchun qo'llaniladi?
45. Nishonlar qo'llanilgan dasturda o'tish operatori qo'llanilmasligi mumkinmi?
46. O'tish operatori qo'llanilgan dasturda nishonlar qo'llanilmasligi mumkinmi?
47. Tarmoqlanish operatori nima uchun qo'llaniladi?
48. Tarmoqlanish operatorida operatorlar ketma-ketligi ishtirok etsa, ular qanday so'lar orasida yoziladi?
49. Tarmoqlanish operatorining qisqa va to'liq ko'rinishlari haqida nimalar bilasiz?
50. Qaysi operatordan keyin nuqtali vergul yozilmaydi?
51. Takrorlanuvchi algoritmlarga misollar keltiring.
52. Parametrlil takrorlash operatorining ko'rinishi qanday?
53. Parametrlil takrorlash operatori qachon qo'llaniladi?
54. Takrorlash parametri qanday qiymatlarni qabul qiladi?
55. Takrorlash parametrining qiymatlari chegaralanganmi?
56. Takrorlash operatorini qo'llanilishini tushuntiring.
57. Qanday holatda Do yoki Downto xizmatchi so'zlari qo'llaniladi?
58. Shart bo'yicha takrorlash operatorining parametrlil takrorlash operatoridan asosiy farqi nimada?
59. Shart bo'yicha takrorlash operatorlardan qaysilarini bilasiz?
60. While operatorining qo'llanilishini tushuntiring.
61. Repeat operatorining qo'llanilishini tushuntiring.
62. Takrorlash operatorlarini qo'llash qulay bo'lgan hollarga mos qilib tushuntirig.
63. Qanday holda takrorlash tanasida begin va end; yozilishi zarur?

Qo'shimcha masalalar

DT-10.1. Uchburchakning tomonlari a , b va ular orasidagi burchak α berilgan. Uchburchakning yuzini hisoblash dasturini tuzing.

DT-10.2. Quyida berilgan shakllarning shtrixlangan sohalarning yuzalarini hisoblash dasturini tuzing.



DT-10.3. Istagan butun sonni 17 ga qoldiqsiz bo'linishi yoki bo'linmasligini aniqlovchi dastur tuzing.

DT-10.4. $ax+b=0$ chiziqli tenglamaning yechish dasturini tuzing.

DT-10.5. Yildagi oylarning kunlari miqdorini aniqlab beruvchi dastur tuzing.

DT-10.6. Oldingi dastur ko'magida tug'ilganingizdan beri necha kun yashaganingizni aniqlang.

DT-10.7. Kvadratlari berilgan natural N sonda katta bo'lmagan natural sonlarni chiqaruvchi dastur tuzing.

DT-10.8. a_1, a_2, \dots, a_N butun sonlar ketma-ketligi berilgan. Ularni ketma-ket qo'shib borib, yig'indi berilgan N butun sonda ortishi bilan ekranga chiqaruvchi dastur tuzing. Agar barcha sonlar yig'indisi N dan ortmasa, bu haqda xabar chiqarilsin.

DT-10.9. $y = x \cdot \sin x$ funksiyaning qiymatlarini $[-\pi, \pi]$ oraliqda 0,3 qadam bilan hisoblash dasturini tuzing.

DT-10.10. Quyidagi yig'indining qiymati berilgan M natural sonda ortiq bo'lguncha hisoblash dasturini tuzing:

$$y = \frac{1}{3} - \frac{1}{10} + \frac{1}{21} - \dots + \frac{(-1)^{N+1}}{N \cdot (2 \cdot N + 1)}.$$

DT-10.11. Avvalgi masalani parametrli shart bo'yicha takrorlash operatorlari yordamida yeching.

DT-10.12. Hisoblash dasturini tuzing:

$$y = \sqrt{2 + \sqrt{4 + \sqrt{6 + \dots + \sqrt{98 + \sqrt{100}}}}}$$

DT-10.13. Hisoblash dasturini tuzing:

$$y = \underbrace{\sqrt{5 + \sqrt{5 + \sqrt{5 + \dots + \sqrt{5}}}}}_n$$

DT-10.14. n natural son berilgan. $p = (1 - 1/2^2)(1 - 1/3^2) \dots (1 - 1/n^2)$ ni hisoblash dasturini tuzing, bunda $n > 2$.

DT-10.15. n natural son berilgan. $P = \frac{1}{2} * \frac{1}{3} * \frac{1}{4} * \dots * \frac{1}{n}$ ko'paytmani hisoblash dasturini tuzing.

DT-10.16. n natural son berilgan. $P = \left(1 + \frac{1}{1^2}\right) \left(1 + \frac{1}{2^2}\right) \dots \left(1 + \frac{1}{n^2}\right)$ ni hisoblovchi dastur tuzing.

DT-10.17. n atural son berilgan. $P = \frac{1}{2} * \frac{3}{4} * \frac{5}{6} * \dots * \frac{2n-1}{2*n}$ ko'paytmani topuvchi dastur tuzing.

DT-10.18. n atural son berilgan. $P = \frac{1}{1} * \frac{3}{2} * \frac{5}{3} * \dots * \frac{2n-1}{n}$ ko'paytmani topuvchi dastur tuzing.

DT-10.19. $P = 1 - \frac{1}{2} + \frac{1}{3} - \dots + \frac{1}{9999} - \frac{1}{10000}$ yig'indini hisoblovchi dastur tuzing.

DT-10.20. N natural son berilgan. Bu sonning barcha tub bo'luvchilarini topuvchi dastur tuzing.

DT-10.21. a, b natural sonlar berilgan ($a \leq b$). $a < p < b$ shartni qanoatlantiradigan barcha tub p sonlarni topuvchi dastur tuzing.

DT-10.22. Birinchi 100 ta tub sonlarni aniqlovchi dastur tuzing.

DT-10.23. N, m natural son berilgan. Raqamlar yig'indisining kvadrati m ga teng va N sondan kichik bo'lgan barcha sonlarni topuvchi dastur tuzing.

DT-10.24. Natural son mukammal son deyiladi, agar bu son o'zining bo'luvchilari yig'indisiga teng bo'lsa. 6 son mukammal son, chunki uning bo'luvchilari 1, 2, 3 ga teng va ularning yig'indisi $6 = 1 + 2 + 3$ ga teng. 8 esa ($8 \neq 1 + 2 + 4$). N natural son berilgan. Ushbu sondan kichik bo'lgan barcha mukammal sonlarni topuvchi dastur tuzing.

DT-10.25. n natural son berilgan. Ushbu sonni uchta natural sonlar kvadratlari yig'indisi orqali ifoda etish mumkinmi ($n = x^2 + y^2 + z^2$). Agar mumkin bo'lsa, u holda $n = x^2 + y^2 + z^2$ tenglik o'rinli bo'ladigan barcha x, y, z sonlar uchligini aniqlovchi dastur tuzing.

FOYDALANILGAN ADABIYOTLAR

1. *Kulakov A.G., Lando S.K., Semyonov A.L., Shen A.X.* Algoritmika. V–VII sinflar. Moskva: Drofa, 1997.
2. *Boltayev B., Abduqodirov A., Taylaqov N., Mahkamov M., Azamatov A., Xafizov S.* Informatika va hisoblash texnikasi asoslari. 9-sinf. T.: Cho'lpon, 2006.
3. *Boltayev B., Mahkamov M., Azamatov A.* Informatikadan olimpiada masalalarini yechish. Metodik qo'llanma, T.: 2004.
4. *Boltayev B., Mahkamov M., Azamatov A.* Informatikadan olimpiada masalalarini yechish-2. Metodik qo'llanma, Toshkent, 2004.
5. *B.Boltayev, A.Abduqodirov, N.Taylaqov, M.Mahkamov, A.Azamatov, S.Xafizov.* Informatika va hisoblash texnikasi asoslari. 9-sinf. T.: Cho'lpon, 2006.
6. *B.Boltayev, Mahkamov M., Azamatov A., Abduqodirov A., Daliyev A., Azlarov T., Taylaqov N.* 8-sinf. T.: O'qituvchi, 2006.
7. *B.Boltayev, M.Mahkamov, A.Azamatov.* 8-sinf masalalar to'plami va ularni yechish usullari. Metodik qo'llanma. T.: 2005.
8. *B.Boltayev, Mahkamov M., Azamatov A.* Paskal dasturlash tili. Metodik qo'llanma. T.: 2007.
9. *B.Boltayev, Mahkamov M., Azamatov A., Rahmonqulova S.* Informatika. 7-sinf. T.: 2006.

MUNDARIJA

I bob. ALGORITMLAR

Algoritm tushunchasi.....	3
Ijrochi.....	6
Algoritmning xossalari.....	9
Algoritmni tasvirlash usullari.....	13
Algoritmning asos turlari.....	16

II bob. IJROCHILAR VA ULARNING KO'RSATMALARI

Algoritmik tillar olamiga kirish.....	21
Ijrochi Dehqon.....	24
Masala o'yin kabi ifodalash.....	26
Ijrochi Suvchi	27
Ijrochi Oshiruvchi.....	29
Ijrochi Chigirtka.....	31
Algoritmni qanday yengillashtirish mumkin.....	33
Ijrochi Baqa.....	36
<i>Qo'shimcha masalalar.....</i>	38
<i>Oshiruvchi uchun.....</i>	41
<i>Baqa uchun.....</i>	47

III bob. PROTSEDURALAR – YANGI KO'RSATMALAR..... 49

<i>Qo'shimcha masalalar.....</i>	53
<i>Chigirtka uchun.....</i>	54

IV bob. TAKRORLANISH TUZILMASI

Askarlar va qayiq.....	56
Yangi tuzilma.....	57
Yangi tuzilmani qo'llash: Chigirtka.....	60
Yangi tuzilmani qo'llash: Oshiruvchi.....	62
Yangi tuzilmani qo'llash: Baqa.....	63
Nima uchun bosh harflar.....	65

Samaradorlik va murakkablik.....	66
Samarador Oshiruvchi.....	69
Oshiruvchi uchun algoritmni yaxshilash.....	72
<i>Qo'shimcha masalalar.....</i>	<i>73</i>

V bob. MANTIQ ELEMENTLARI VA SHARTLAR

Mantiq elementlari.....	75
Qiziqarli mantiqiy masalalar.....	78
Takomillashtirilgan Kamaytiruvchi.....	80
Tarmoqlanuvchi algoritmlar blok-sxemalari.....	84

VI bob. IJROCHI ROBOT

Robot qayerda yashaydi?.....	93
Robotning imkoniyatlari bilan tanishuv.....	95
Takrorlanuvchi holatlar.....	104
Qiziqarli masalalar.....	111
Birikkan shartlar.....	117
Birikkan shartli algoritmlar.....	118
Murakkab algoritm namunasi.....	122
<i>Qo'shimcha masalalar.....</i>	<i>125</i>

VII bob. REKURSIYA

O'zini chaqiradigan protseduralar.....	129
Rekursiyadan chiqish.....	133
Qachon rekursiyasiz ishlash mumkin.....	134
Oraliq protsedura orqali rekursiv chaqirish.....	136
<i>Qo'shimcha masalalar.....</i>	<i>137</i>

VIII bob. SARALOVCHI UCHUN MASALALAR

Saralovchi tarixi.....	139
O'tkazish masalalari.....	141
Tanlash masalalari.....	146
Yangi Saralovchi sari.....	152
Ichma-ich joylashgan sikllar.....	154
Saralash usullari.....	157
<i>Qo'shimcha masalalar.....</i>	<i>167</i>

IX bob. MUKAMMAL SARALOVCHI

Yangi imkoniyatlar.....	170
Sodda masalalar.....	172
Qiyinroq masalalar.....	179
Qo'shimcha masalalar.....	185
<i>Qo'shimcha masalalar.....</i>	<i>192</i>

X bob. DASTURLASH ASOSLARI

Tokchalar uchun ba'zi chegaralashlar.....	195
Dastur va dasturlash tillari.....	197
Ijrochilar va dasturlash tillari.....	201
O'zgaruvchi miqdorlar.....	203
Jadval ko'rinishidagi miqdorlar.....	206
Ba'zi standart funksiyalar va algebraik ifodalar.....	210
O'zlashtirish va ma'lumotlarni ekranga chiqarish operatorlari.....	212
Ma'lumotlarni xotiraga muloqot usulida kiritish operatori.....	213
Chiziqli dasturlar tuzish.....	214
O'tish va tarmoqlanish operatorlari.....	216
Takrorlash operatorlari.....	220
<i>Qo'shimcha masalalar.....</i>	<i>225</i>
Foydalanilgan adabiyotlar.....	228

13 YAAT - 21

AXAT RAXMATOVICH AZAMATOV

**ALGORITMLASH VA
DASTURLASH ASOSLARI**

Kasb-hunar kollejlari uchun o'quv qo'llanma

Ikkinchi nashri

Muharrir Xudoyberdi Po'latxo'jayev

Badiiy muharrir Alyona Tulkinova

Texnik muharrir Yelena Tolochko

Musahhah Ozoda Axmedova

Kompyuterda sahifalovchi Gulbayra Yeraliyeva

Litlenziya raqami AI № 163. Bosishga ruxsat etildi 22.09.2011. Bichimi 60×90¹/₁₆, Tayms TAD garniturası. Shartlı b.t. 14,5. Nashr b.t. 14,35. Shartnoma № 82–2011. 1240 nusxada. Buyurtma № 718.

O'zbekiston Matbuot va axborot agentligining Cho'lpon nomidagi nashriyot-matbaa ijodiy uyi. 100129, Toshkent. Navoiy ko'chasi, 30- uy.

«Shoakbar» MCHJ bosmaxonasida chop etildi. 100031, Toshkent, To'g'on Rejimetov ko'chasi, 1 a.



Cholpon



**Cho'lon nomidagi
nashriyot-matbaa ijodiy uyi**

ISBN 978-9943-05-439-4



9 789943 054394