O’ZBEKISTON RESPUBLIKASI

OLIY VA O’RTA MAXSUS TA’LIM VAZIRLIGI

GULISTON DAVLAT UNIVERSITETI

FIZIKA-MATEMATIKA FAKULTETI

“Axborot texnologiyalari” kafedrasi

5110700- “Informatika o’qitish metodikasi” ta’lim yo’nalishi

5-16 guruh talabasi **Qayumov Sanjar Ilhom o’g’lining**

“Dasturlashga oid olimpiada masalalari va ularni yechish metodikasi” mavzusida bajargan.

BITIRUV MALAKAVIY ISHI

Rahbar: f.-m.f.Ph.D. A.A.Qalandarov

Guliston - 2020

Bitiruv malakaviy ish Guliston davlat universitetining 2020 yil _____ dagi _____-sonli buyrug'i bilan tasdiqlangan Davlat attestatsiya komissiyasining ____–sonli yig'ilishida muhokama qilindi va "____" ball bilan (_____) baholandi.

Bitiruv malakaviy ish "Fizika-matematika" fakultetining 2020 yil "____"_____dagi ____–sonli Ilmiy-uslubiy kengashi qarori bilan Davlat attestatsiyasi komissiyasiga himoya qilish uchun tavsiya etildi.

Fakultet dekani S.Allayorov

Bitiruv malakaviy ish "Axborot texnologiyalari" kafedrasining 20___ yil "____"_____dagi ____– sonli yig'ilishida muhokama qilindi va himoyaga tavsiya etildi.

Kafedra mudiri A.Qalandarov

BMI bajaruvchi "5110700- Informatika o'qitish metodikasi" ta'lim yo'nalishi 5-16-guruh talabasi: _____ S.I.Qayumov.

Rahbar: _____ f.-m.f.Ph.D. A.A.Qalandarov

Mundarija

Kirish

Ma'lumki, joriy yil yurtimizda “Ilm - ma'rifat va raqamli iqtisodiyotni rivojlantirish yili”, deb e'lon qilindi. Barcha sohalarda ochiq-oshkoralik va samaradorlikni ta'minlash maqsadida “Elektron hukumat” tizimi keng joriy etilmoqda.

Jamiyat taraqqiyotida, yurt, millat taqdirida o‘sib kelayotgan yosh avlodning jismoniy va ma’naviy barkamolligi, intellektual salohiyati muhim o‘rin tutadi. Zero, davlatimiz rahbari Shavkat Mirziyoyev ta’kidlaganidek, “Dunyo shiddat bilan o‘zgarib, barqarorlik va xalqlarning mustahkam rivojlanishiga raxna soladigan turli yangi tahdid va xavflar paydo bo‘layotgan bugungi kunda ma’navivat va ma’rifatga, axloqiy tarbiya, yoshlarning bilim olish, kamolga yetishga intilishiga e’tibor qaratish har qachongidan ham muhimdir”.

O’zbekiston Respublikasini rivojlantirishning beshta ustuvor yo‘nalishi bo‘yicha Harakatlar strategiyasida belgilangan yoshlarga oid davlat siyosatini takomillashtirish, yoshlarning huquq va manfaatlarini himoya qilish, ularning barkamol bo‘lib voyaga yetishi uchun zarur shart-sharoitlar yaratish borasida bir qancha muhim hujjatlar qabul qilindi, amaliy ishlar yo‘lga qo‘yildi, muayyan natijalarga erishildi. Shunday bo‘lsa ham, bu borada bajariladigan ishlar hamisha keng ko‘lamli bo‘lib, o‘z dolzarbligini saqlab qolaveradi. [4]

O’zbekiston Respublikasi Prezidenti Sh.M.Mirziyoyevning 2019-yil 19-martda “Yoshlar bilan ishlashni samarali tashkil etishda madaniyat, san’at, sport, axborot texnologiyalari, kitob o‘qishga qiziqishini oshirish bo‘yicha 5 ta muhim tashabbusni amalga oshirish to‘g‘risida” o‘tkazilgan videoselektor yig'ilishida yoshlarni rivojlantirishga qaratilgan masalalar muhokama qilindi va sohalarga taalluqli 5 ta muhim tashabbus ilgari surildi.

Bular dan ***uchinchi tashabbus*** – “Aholi va yoshlar o‘rtasida kompyuter texnologiyalari va internetdan samarali foydalanish” masalalari bo‘yicha. Prezidentimiz Sh.M.Mirziyoyev Axborot texnologiya sohasiga katta e’tibor qaratmoqda. Hususan mamlakatimizda dasturlash sohasini yanada rivojlantirish,

bo’yicha turli xil ishlar amalga oshirilmoqda.

“Bugun O‘zbekiston barcha sohalarda o‘zini namoyon etmoqda. O‘z vaqtida axborot texnologiyalariga e’tibor qaratganimiz yaxshi natijalar bermoqda. Bitta dasturiy mahsulot davlatimizga qancha naf keltiradi, korrupsiya, byurokratiyani bartaraf etadi, odamlarga qulaylik yaratadi. Har doim yodda tutinglarki, sizlarning ishingiz judayam muhim” , - dedi Prezidentimiz Sh.M.Mirziyoyev.

O’zbekiston Respublikasi Prezidenti Shavkat Mirziyoyev Oliy Majlisga yo'llagan Murojaatnomasida Toshkent shahrida zamonaviy infratuzilmaga ega bo‘lgan Dasturiy mahsulotlar va axborot texnologiyalari texnologik parki «IT-park» barpo etilayotganini, u hozirdanoq o'zining dastlabki natijalarini bera boshlaganini ta’kidlagandi.

Shuningdek, bunday «IT-park»lar Nukus, Buxoro, Namangan, Samarqand, Guliston va Urganch shaharlarida ham Soha uchun yuqori malakali mutaxassislar tayyorlash maqsadida xorijiy hamkorlarimiz bilan birgalikda «Bir million dasturchi» loyihasini amalga oshirish boshlangani alohida e’tirof etildi.

«Bir million o’zbek dasturchilari» - Loyiha maqsadi, o’zbek tilidagi onlayn platforma kurslari orqali o’zbek yoshlariga axborot texnologiyalarini o’rgatish, yosh dasturchilar yetishtirishdir. Onlayn platformada 4 ta’lim yo’nalishi bo’yicha kurslar tashkil etiladi, bular:

- Android dasturlash;
- Full – Stack Development;
- Frontend Development;
- Ma’lumotlar tahlili.

Bu kurslarda o’qish 120 soatni tashkil etilishi rejalashtirilgan bo’lib, har bitta bosqich yakunida o’quvchilar uy ishlari va labaratoriya ishlarini bajarib, onlayn topshirishadi. Va bunda ularga maxsus sertifikatlar topshiriladi. Prezidentimiz Shavkat Miromonovich Mirziyoyev dasturlash sohasiga juda katta e’tibor qaratmoqda.

Mustaqil rivojlanish yo'lidan borayotgan Respublikamizda yangi iqtisodiy sharoitlar ro'yobga keldiki, bu xalq xo'jaligining barcha sohalarida, hususan, ta'lim

tizimida ham qator islohotlarni amalga oshirishni taqozo etadi. Fan va texnika taraqqiyotining jadal kechayotgani, kompyuter texnikasining keng qo'llanilayotganligi, o'quvchilarga berilishi lozim bo'lgan bilimlar ko'lamining tobora oshib borayotganligi, ta'limni tubdan yaxshilashni, fanlar bo'yicha milliy va mahalliy sharoitlarga mos keladigan hamda malakali mutahassislarni tayyorlash imkonini beradigan dastur va darsliklar, ta'limning zamonaviy metodlarini ishlab chiqishni talab qiladi.

O'zbekiston Respublikasi Oliy Majlisi qabul qilgan «Ta'lim to'g'risida»gi qonunda ta'limni yanada takomillashtirish maqsadida Respublika hududida ta'limni umumiy o'rta ta'lim maktablari, akademik litseylar va kasb-hunar kollejlari orqali amalga oshirish maqsad qilib qo'yilgan bo'lsa, «Kadrlar tayyorlash milliy dasturi» qonunida milliy tajribalarni tahlil qilgan holda hamda jahonning ilg'or yutuqlarini e'tiborga olib akademik litsey va kasb-hunar kollejlarda har tomonlama shakllangan va yetuk mutaxassislarni tayyorlashning asosiy yo'nalishlari belgilab berilgan.

Mavzuning dolzarbligi: Hozirgi kunda o'quvchi yoshlarning Informatika fanlaridan olimpiadalari, mamlakatimizda katta taraqqiyot yo'lini bosib o'tmoqda.

Informatikadan turli bosqichlarda (maktab, tuman, shahar, viloyat, respublika va xalqaro) olimpiadalar o'tkazish, informatikani o'zlashtirish darajasini aniqlashning samarali vositasi va yoshlarning informatika va dasturlash fanlarini tushunish darajasini hamda ularning ijodiy imkoniyatlarini nazort qilish uchun ishonchli vositadir .

Olimpiadalar - teng kuchli o'quvchilar orasidan eng, kuchlisini aniqlash maqsadida musobaqalar o'tkazishning an'anaviy shaklidir. Ko'pgina informatika o'qituvchilari , ayniqsa yosh o'qituvchilar va turli bosqichdagi olimpiadalarning tashkilotchilari topshiriqlarni tayyorlashda va olimpiadalarni o'tkazishda, eng muhimi , qobilyatli o'quvchilarni bunday musobaqalarga tayyorlash jarayonida

tegishli metodik adabiyot tanlashda hamda mashg'ulotlar tashkil qilishda anchagina qiyinchiliklarga uchraydilar.

Biror masalaning qiyinlik darajasi to'g'risidagi fikr, odatda, judayam subyektivdir. Shu sababli dasturlashga oid olimpiada masalalarini yechish usullari o'quvchining qobiliyatiga juda bog'liqdir.

Tanlangan masalalar orasida nisbatan murakkablari ham uchraydi. Bunday masalalar chuqur bilim bilan birga topqirlikni, odatdagicha bo'lmagan yoki murakkab vaziyatlarni anglab yetish uquvini talab qiladi. Masalalarning qiyinligi har xil bo'ladi; ba'zilarining mantiqiy mazmuni elementar bo'lsa ham, ularni yechish uchun murakkab matematik hisoblashlar bajarish talab etiladi va aksincha, ayrim masalalarning algoritmik mohiyati birmuncha murakkab bo'lsa ham, uni

yechishning ratsional usuli tanlansa, ular osongina yechiladi. Masalalar yechish — ilmiy bilimlar va tushunchalar sistemasini egallash vositalaridan biridir. Bilish va amaliy xarakterdagi o'quv va malakalarni egallashda masalalar yechishning roli juda kattadir. Olimpida masalalarini yechish o'quvchilarning umumiy aqliy rivojlanishlarida muhim ahamiyatga egadir.

Tadqiqot maqsadi: Talaba yoshlarda dasturlash ko'nikmasi va malakasini shakllantirish, o'rta va murakkab darajadagi olimpiada masalalarini yechish bosqichlari, turli murakkablikdagi masalalarni har xil usul va metodlardan foydalanib yechishning ilmiy uslubiy asoslarini ishlab chiqish, dasturlar tuzishni ijodiy- tadqiqot darajasiga ko'tarish, mustaqil fikrlashni rivojlantirish, talabalarga Algoritmlar va dasturlash tillarida muammoli va murakkab masalalarni algoritmalshtirish, modellashtirish va dasturlashni o'rgatish eng asosiy maqsaddir.

Tadqiqot vazifalari: Dasturlashga oid olimpiada masalalari va ularni yechish metodikasi ning xar bir masalasi uchun tayanch muammolarni yuzaga chiqarib, algoritmini tuzish, muqobil yechim yo'llarini izlab topish va optimal yechimlarni tanlash va dasturlarni shakllantirish strukturasi.

Dasturlashga oid olimpiada masalalarini yechish usullari, o’rganilayotgan sohanining ilmiy yutuqlaridan kelib chiqqan holda yechish, o’quv samaradorligini oshirish ushbu bitiruv malakaviy ishining asosiy vazifalaridan biridir.

Zamonaviy axborot texnologiyalarning nazariy asoslari va printsiplarini o’rganib, kasb ta’limi fanlariga jumladan, “Algoritmlar va dasturlash tillari” fanlariga tadbiq etishni o’rganish bilan o’quv samaradorligini oshirish sohasidagi ilmiy tadqiqot ishlarini olib borish masalasi qo’yilgan. Masalalarni yechishda tanlangan metodlar ichida eng optimal yechimni tanlab, shu masalalarni ishlash usulini to’liq yoritib berish, amaliyotda joriy qilish, ko’nikma va malakalarini shakllantirishdan iborat.

Tadqiqot ob’ekti va predmeti:

Ob’ekti: Informatika va axborot ta’lim yo`nalishi bo`yicha kadrlar tayyorlovchi maktablar, akademik litseylar va kasb-hunar kollejlari.

Predmeti: Masalalarning qo’yilishi, algoritmlashtirish, dasturlash masalalarni turli xil metodlardan foydalanib yechish mohiyatini o`rganish hamda uni nazariy va amaliy tadqiq etish hisoblanadi.

Kutiladigan natija: O’quvchilar informatika fanidan dasturlashga oid olimpiada masalalarini yechish jarayonida algoritm asosida yotgan fundamental qonunlar bilan chuqurroq tanishadilar, turli hodisalarni ilmiy tekshirishning metodlarini egallaydilar. Masala yechishning tarbiyaviy ahamiyati shundaki, bunda o’quvchilarda mehnatsevarlik, izchillik, o’z erkini qo’lga olish va maqsadga intilish kabi xususiyatlar shakllanib boradi.

Tadqiqot metodlari: Matematik modellashtirish, algoritmlar nazariyasi va dasturlash

Tadqiqotning ilmiy yangiligi va amaliy ahamiyati: Bitiruv malakaviy ishida Dasturlashga oid olimpiada masalalari yechishda zamonaviy usul va metodlardan foydalanib, nazariy-ilmiy asoslab berildi.

Foydalanilgan manba va adabiyotlarning tanqidiy tahlil: Ushbu bitiruv malakaviy ishni bajarish mobaynida mavzuga tegishli bir qancha qo’llanma, o’quv adabiyotlari, internet sahifalari materiallari hamda ilmiy - nazariy anjuman materiallari o’rganib chiqildi, foydalanilgan va o’rganilgan adabiyotlar tahlil qilingan holda bitiruv malakaviy ishi tayyorlandi. Ular bitiruv malakaviy ishi so’ngida keltirilgan adabiyotlar ro’yxatida aks etgan.

Ishning tarkibiy tuzilishi. Bitiruv malakaviy ishi kirish, ikkita bob, xulosa va foydalanilgan adabiyotlardan tashkil topgan. Birinchi bobida Berilgan masalaning qo’yilishi va uning algoritmini tuzish” to’g’risida ma’lumot berilgan.

Ikkinchi bobda “Dasturlash uslubiyatlari va masalani yechish metodlari” haqida yoritib berilgan.

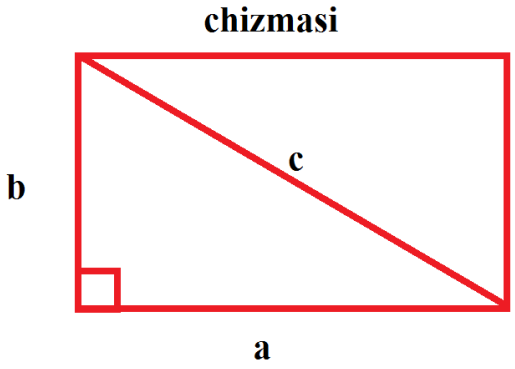
I. BOB Berilgan masalaning qo’yilishi va uning algoritmini tuzish

1.1 Masalaning qo’yilishi va undan kutilayotgan natija

Inson hayoti davomida juda ko’p masalalarni yechishga to’g’ri keladi. Bu masalalarning bazilari oson, bazilari murakkab hisob-kitoblar bilan bog’liq bo’ladi. Ba’zi bir masalalarni yechishda biror amallar ketma-ketligi minglab marotaba bajarilishiga to’g’ri kelishi mumkin. Bu masalalarni kompyuterda hal etish mumkinmi degan savol paydo bo’ladi albatta. Agar kompyuterda bajara olsa, u holda masalalarni kompyuterda hal etish qanday amalga oshiriladi, degan savol tabiiydir.

Bu savolga javob berishdan avval bir nechta masalalarni va ularning yechish usullarini ko’rib o’tamiz.

Masala-1: Qizil rangli qalamda chizilgan To’g’ri burchakli to’rtburchakning tomonlari, mos ravishda, 4sm va 3sm bo’lsa, uning diagonali uzunligini toping (Yo’llanma: to’g’ri to’rtburchakning diagonali to’rtburchakni ikkita to’g’ri burchakli uchburchakka ajratadi, demak diagonal, gipotenuza bo’ladi).



Masalani tahlil etamiz: Birinchidan masalaning yechimini topish uchun to’g’ri to’rtburchakning qanday rangli qalamda chizilganligining ahamiyati yo’qligini, ya’ni biz uchun ***“keraksiz”*** axborot, ikkinchidan to’rtburchakning ***to’g’ri burchakli*** bo’lishi muhim axborot ekanligini aniqlaymiz.

Masalaning qo’yilishi.

Berilgan;

To’rtburchak to’g’ri burchakli, demak diagonali to’rtburchakni ikkita to’g’ri burchakli uchburchakka ajratadi.

a=4 sm

b=3 sm

Topish kerak: c diagonal Masalaning modelini tuzish.

Pifagor teoremasi: $c^2=a^2+b^2$;

Algoritm tuzish. (Bajarilayotgan amallar aniq va tushunarli bo'lishi uchun AA, BB, CC kabi belgilashlar kiritib olamiz)

1) AA=a*a;

2) BB=b*b;

3) CC=AA+BB

4) c=√CC

Natija olish va tahlil qilish

1) AA=4sm*4sm=$16sm^2$

2) BB=3sm*3sm=$9sm^2$

3) CC=$16sm^2+9sm^2=25sm^2$

4) c=√CC=√$25sm^2$=5sm

Natijaning tahlili:

$(5sm)^2=(4sm)^2+(3sm)^2$ yoki $25sm^2=16sm^2+9sm^2$

Demak, natija to'g'ri.

Javob: c=5sm.

Masala-2: Behzod kitobning to'rt sahifasini va yana to'rtta satrini o'qidi. Kitob sahifasida qancha satr bo'lsa, har bir satrda shunchadan belgi mavjud. Agar Behzod o'qigan axborot 6560 bayt bo'lsa, kitobning bir sahifasida nechta satr borligini aniqlang?

Masalani tahlil qilishga o’tamiz.

Masalaning boshlang’ich qiymatlari:

- Behzod kitobning sahifasi va 4 satrini o’qigan;
- Behzod o’qigan axborot 6560 bayt;
- Sahifadagi satrlar soni satrlardagi belgilar soniga teng.

Masalaning maqsadi.

Kitob sahifasida nechta satr borligini aniqlash.

Masala shartlariga mos tenglama tuzish

Masalada topish talab etilgan satrlar sonini x bilan belgilaymiz. U holda shartga ko’ra har bir satrda x tadan belgi bo’ladi.

Demak, kitobning bitta sahifasida x^2 ta (x ta belgidan iborat x ta satr) belgi bor. Masala shartiga ko’ra Behzod $4x^2+4x$ ta (4 ta sahifa va 4 ta satr) belgi o’qigan. Masala shartiga asosan bu belgilarning soni 6560 bayt (bitta belgi – bir bayt) ga teng:1

$4x^2+4x=6560$

Tenglamani $x^2+x-1640=0$ ko’rinishdagi kvadrat tenglamaga keltiramiz, ya’ni masalaning shartlariga mos tenglama hosil qildik.

Tenglamani yechish ketma-ketligi:

Sizga ma’lum bo’lgan kvadrat tenglama yechish usulidan foydalaniladi:

1) Diskirminant hisoblanadi: $D=1^2-4*1*(-1640)=6561=81^2$

2) D>0 bo’lgani uchun ikkita yechim topiladi:

$X_1=(-1-81)/2*1= - 41;$ $X_2=(-1+81)/2*1=40;$

Natijaning tahlili :

Tenglamaning ikkita yechimi bor ekan. Lekin kitob sahifalarining soni manfiy bo'la olmaydi, ya'ni tenglamaning masalani qanoatlantiradigan yechimi x=40 ekan. **Javob: 40** ta satr. [8]

Yuqoridagi masalalarning yechilishini tahlil qilib, ular quyidagi bosqichlardan iborat ekanligini ko'rish mumkin:

1. Har bir masalada avval masalaning qo'yilishi, ya'ni masalada berilgan boshlang'ich qiymatlar va masalaning maqsadi (topilishi kerak bo'lgan natijaviy miqdorlar) aniqlanadi.

2. Masalani yechish uchun zarur bo'lgan formulalar, boshqacha aytganda matematik munosabatlar hosil qilinadi.

3. Masala yechimidagi amallar (formulalar, munosabatlar)ni bajarish ketma-ketligi aniqlanadi.

4. Natija olish va tahlil etish.

Yuqoridagi kabi boshqa masalalarni ham kompyuter yordamida hal etish mumkin va u yuqoridagi 4 bosqichga qo'shimcha amallarni kompyuter tushunadigan tilga o'girish va kompyuter xotirasiga kiritish kabi bosqichlarni o'z ichiga oladi:

1) Masalaning qo'yilishi – Masalaga mos boshlang'ich qiymatlar va natijaviy miqdorlar aniqlanadi.

2) Masalaning modelini tuzish – Masala ko'rilayotgan sohaning ilmiy yutuqlaridan kelib chiqib, formulalar orqali ifodalanadi.

3) Algoritm tuzish – Masalaning modelidan foydalanib, hal etishning ko'rsatmalar ketma-ketligi tuziladi.

4) Dastur tuzish - Algoritmdagi ko'rsatmalar ketma-ketligini kompyuter tushunadigan tilga o'tkaziladi.

5) Dasturni kompyuter xotirasiga kiritish – Tuzilgan dastur kompyuter xotirasiga kiritiladi.

6) Natija olish va uni tahlil etish – Dastur ishlatiladi va natijasi tahlil qilingach, xato va kamchiliklar bartaraf etiladi.

Masalaning qo'yilishi va undan kutilayotgan natija – ya'ni biz yechimini topayotgan masalamizdan kutilayotgan natija albatta tushunarli, ishonchli va foydalanuvchi uchun batafsil ma'lumotlarga ega bo'lishi kerak. Masalani turli xil usullardan foydalanib yechishimiz mumkin. masalan matematika faniga oid $(a+b)^2$ yig'indining kvadrati qisqa ko'paytirish formulalarini qo'llash orqali biz masalaning yoki misolning yechimini darrov topishimiz mumkin. agar qisqa ko'paytirish formulasini bilmasak ham ularni bir-biriga ko'paytirish orqali $(a+b)(a+b)=a^2+ab+ab+b^2=a^2+2ab+b^2$ misolning javobini topishimiz mumkin. ikkala usulda ham bir xil natija chiqadi, lekin 1-usul ya'ni yig'indining kvadrati qisqa ko'paytirish formulasi yordamida natijani tezkorlik bilan ko'rishimiz mumkin, bu esa dasturchining bilim darajasi va aqliy salohiyati yuksakligidan dalolat beradi. Yuqorida aytganimizdek masalani yechish metodikasi turli xil bo'lishi mumkin va ular bir xil natijani ham berishi mumkin. Lekin o'quvchi qaysi masalani qanday usulda yechish kerakligini bilishi kerak. Olimpiyada masalalari boshqa masalalarga nisbatan murakkabroq bo'lib, Masalani yechishda uning o'ziga xos tartibli ketma-ketliklar orqali bajarilishi maqsadga muvofiq, va aniq isbotlar, formulalar, teoremalar asosida bajarilib to'g'ri natijaga olib keladi. Dastulashga oid olimpiada masalalari yechishning turli xil metodlari bor, masalan birorta masalani dasturlash tilida yechish uchun u masalani algoritmi imkon qadar sodda va kam harakatlar (amallar) bajarib, kompyuter xotirasidan kam joy egallashi kerak va natija to'g'ri va tushunarli bo'lishi kerak.

Masalani yechishdan oldin, uni berilishini aniq shakllantirib olish zarur. Bu jarayon to'g'ri savollarni aniqlash bo'lib, savollar quyidagicha bo'lishi mumkin:

1.1. Dastlabki berilgan masala shartlarida hamma iboralar tushunarlimi?

1.2. Nima berilgan?

1.3. Nimani topish kerak?

1.4. Yechimni qanday ta’riflash kerak?

1.5. Qaysi berilganlar yetarli emas va hammasi kerakmi?

1.6. Qanaqa mumkinliklar qabul qilingan?

Albatta, bulardan tashqari boshqa savollarni ham ishlatish mumkin, yoki ayrim savollarni bir necha bor takror ishlatishga to’g’ri keladi.

Masalaning qo’yilishi va uning o’ziga xos jihatlari haqida yuqorida aytib o’tdik. Masalaning ishlanish yo’llari bir necha xil usullarda amalga oshirish mumkin. lekin masalaning to’g’ri qo’yilishi, uning algoritmi to’g’ri tuzilishi va kam vaqt ichida, eng yuqori natijaga erishish lozim. Masalaning algoritmi to’g’ri. qulay va ixcham, va tushunarli tuzilsa, uni ixtiyoriy dasturlash tiliga oson kiritish mumkin va bu dasturlash tillarida xech bir muammosiz to’g’ri va aniq ishlaydi. [18]

1.2 Masalani yechish algoritmini ishlab chiqish

Har bir inson xayotida sodda yoki murakkab bo’lgan ko’plab masalalar uchrab turadi. Bu masalalarni ma’lum qoida va instruktsiyalarga asoslangan xolda yechish mumkin. Ko’pgina masalalarni yechishni inson texnik qurilmalar-avtomatlar, EHM, robotlarga topshirishi mumkin. Ikkala xolda ham qo’yilgan masalani yechish uchun, avval uning algoritmini tuzish zarur. Buning moxiyati shundan iboratki, agar algoritm ishlab chiqilgan bo’lsa, uni yechilayotgan masala bilan tanish bo’lmagan biron bir ijrochiga, shu jumladan kompyuterga xam bajarish uchun topshirsa bo’ladi va u algoritmning qoidalariga aniq rioya qilib masalani yechadi.

Algoritm deganda biror maqsadga erishishga yo’naltirilgan, ijrochi bajarishi uchun mo’ljallangan buyruqlarning tartibli ketma-ketligi tushuniladi.

Algoritm so’zi, arifmetik amallarni bajarish qoidalarini bayon qilgan, IX asrning buyuk matematigi Al-Xorazmiy nomining lotincha shaklidan kelib chiqqan. Dastavval algoritmlar deganda ko’p xonali sonlar bilan to’rt arifmetik amal bajariladigan qoidalar tushinilar edi. Keyinchalik bu tushuncha qo’yilgan masalani yechishga olib keladigan qoida va amallar ketma-ketligini belgilash uchun qo’llanila boshladi. Odatda algoritmlardagi ko’rsatmalar ijrochiga tushunarli bo’lishi uchun sodda amallardan iborat bo’lishi kerak.

Algoritm ijrochisi – algoritmdagi ko’rsatilgan buyruq yoki ko’rsatmalarni bajara oladigan abstrakt yoki real (texnik yoki biologik) sistema.

Algoritmning asosiy xossalari:

1. Tushunarlilik. Algoritm ijrochiga tushunarli bo’lishi uchun ijrochining imkoniyatlarini bilishi 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, electron soat, dastgohlar) bo’lsa, u holda algoritm shu texnik vositaning imkoniyatlaridan kelib chiqib tuzilishi kerak.

Demak, berilayotgan har qanday ko’rsatma ijrochining ko’rsatmalar sistemasidan olinishi, ya’ni ijrochi uni qanday bajarishni bilishi kerak ekan.

2. Aniqlik. Algoritmdagi barcha amallar, ko‘rsatmalar yoki buyruqlar bir ma’noli va aniq bo‘lishi kerak. Masalan, «keragicha suv quyilsin» (kerak deganda qancha suv nazarda tutildi: 1 litrmi, 100 litrmi, 1 tonnami?), «insho yozib kelinsin» (qaysi mavzuga oid?) kabi ko‘rsatmalar har xil (ko‘pincha keraksiz) natijalarga olib keladi.

Bundan shunday xulosaga kelamiz, aniqlik xossasiga asosan algoritm ijrochisi ko‘rsatmalar ketma-ketligini mexanik ravishda xatosiz bajaradi va qo‘shimcha izohlar talab qilmaydi.

3. Diskretlilik (uzluklilik, alohidalik). Algoritmda masalani yechish jarayoni alohida olingan sodda ko’rsatmalar ketma-ketligini qadamma-qadam bajarishdan iborat bo’lishi kerak. Bu xossa avvalgi masalada yaqqol ko’rinib turibdi.

4. Natijaviylik (cheklilik). Algoritm masalaning yechimiga chekli sondagi qadamlar ichida olib kelishi yoki masalani “yechib bo’lmaydi” degan xabar bilan tugatishi kerak.

Bu xossaning mazmuni shundan iboratki, har qanday algoritm ijrosi 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 ham mumkin. Ikkinchi tomondan, qo‘yilgan masala ijobiy yechimga ega bo‘lmasligi ham mumkin. Lekin salbiy natija ham natija deb qabul qilinadi.

5. Ommaviylik. Biror masalani yechish algoritmi umumiy hollar uchun tuziladi, ya’ni faqatgina boshlang‘ich ma’lumotlar bilan farqlanuvchi bir turdagi masalalar turkumi uchun tuziladi. Yuqoridagi «$ax^2+bx+c=0$ $(a\neq0)$ ko’rinishidagi kvadrat tenglamani yechish » algoritmi ixtiyoriy a, b, c sonlar uchun natija beradi, ya’ni algoritmning ommaviylik xossasi o’rinlidir.

Algoritmni tasvirlash usullari

Algoritmning so’zlar yordamida ifodalanishi.

Algoritmning formulalar yordamida ifodalanishi.

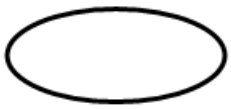
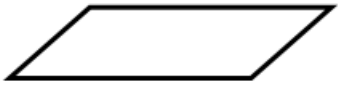
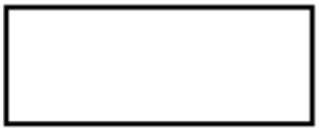
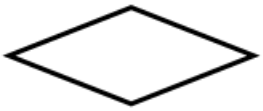
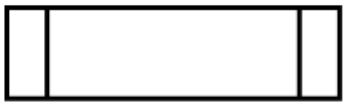
Algoritmning jadval yordamida ifodalanishi..

Algoritmning grafik shaklda ifodalanishi.

Algoritmning dastur shaklida ifodalanishi.

Algoritmning grafik shaklda ifodalanishi.

Algoritmlash asoslarini o’rganishning yana bir qulay grafik shakli blok-sxema usulidir. Blok-sxemalar yo’nalish chiziqlari orqali tutashtirilgan ma’lum buyruq yoki ko’rsatmani aks ettiruvchi maxsus geometrik shakllar – bloklardan tashkil topadi:

	Algoritmning boshlanishini va tugallanganligini bildiradi
	Ma’lumotlarni kiritish va chiqarishni bildiradi.
	Oddiy harakatni, ya’ni qiymat berish yoki tegishli ko’rsatmalar berishni bildiriadi.
	Shart tekshirilishini bildiradi.
	Yordamchi algoritmga murojaatni bildiradi.
	Sxemadagi harakat yo’nalishini bildiradi
:=	Qiymat berish ko’rsatmasi.

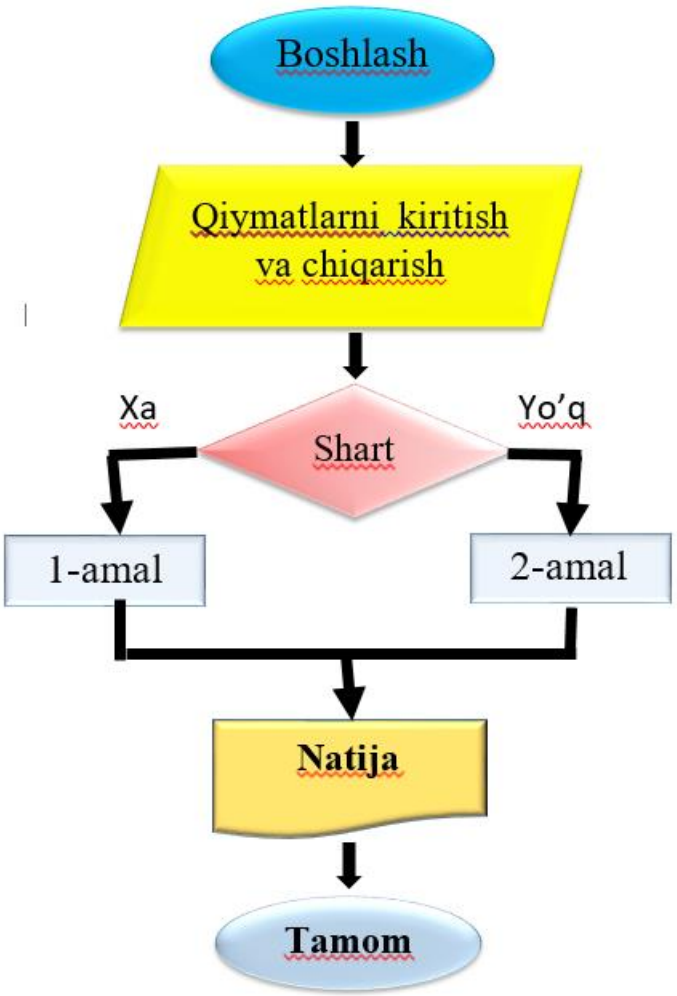
Algoritmning asosiy turlari

Har qanday algoritm mantiqiy tuzilishga, ya’ni bajarilish tartibiga qarab uch asosiy turga bo‘linadi:

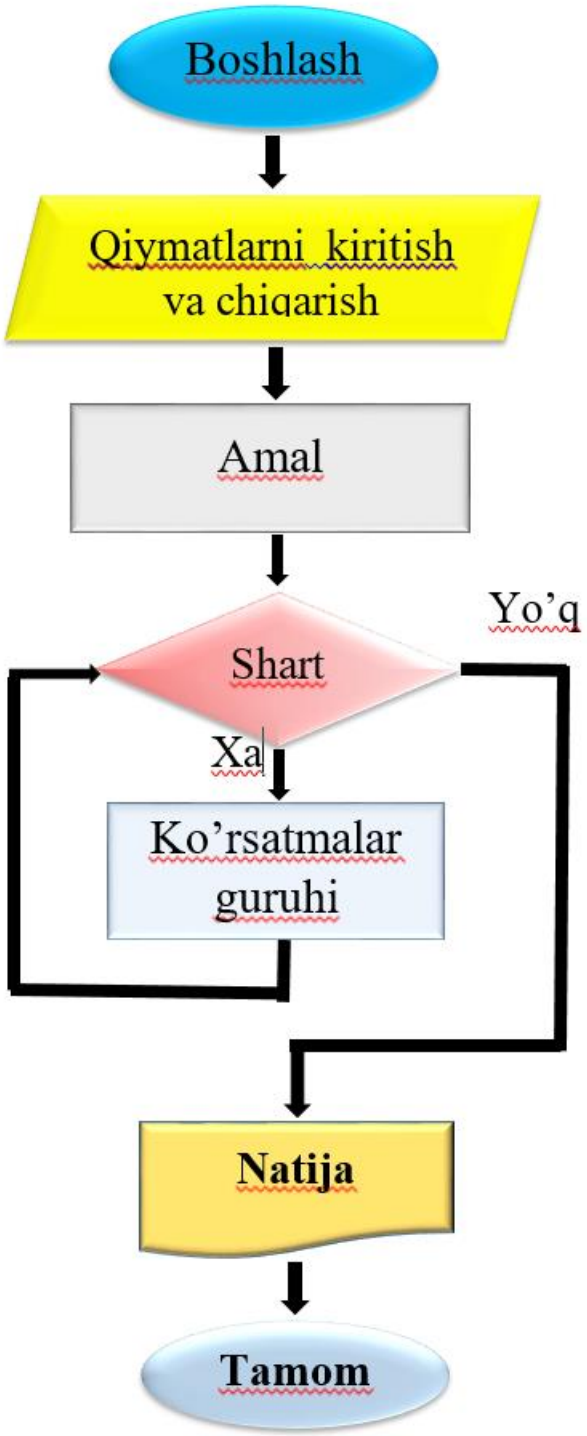
- Chiziqli,
- Tarmoqlanuvchi,
- Takrorlanuvchi.

Chiziqli , tarmoqlanuvchi va takrorlanuvchi algoritmlarning blok-sxemalar orqali ifodalanishiga namuna

	**Chiziqli algoritmlar**. Barcha ko‘rsatmalari ketma-ket joylashish tartibida bajarib boriladigan algoritmlar chiziqli algoritmlar deyiladi. «ikkita sonning yig’indisini 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. Algoritmlarning bu turi hayotimizda har kuni va har qadamda uchraydi. Eshikdan chiqishimiz eshik ochiq yoki yopiqligiga, ko‘chaga kiyinib chiqishimiz ob-havoga, biror joyga borish uchun transport vositasini tanlashimiz to‘lash imkonimiz bo‘lgan pulga bog‘liqdir. Demak, tarmoqlanuvchi algoritmlar chiziqli algoritmlardan tanlanish imkoniyati bilan farqlanar ekan.



Takrorlanuvchi (siklik) algoritmlar. Masalalarni tahlil etish jarayonida algoritmdagi ba'zi ko'rsatmalar takroran bajarilishini kuzatish mumkin. Hayotimizda juda ko'p jarayonlar takrorlanadi. Masalan, darslarning takrorlanishi, … 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. Bunday ko'rsatmalarning mazmunini tushunish uchun takrorlanishning bir nechta qadamini ko'rib chiqish lozim.

Algoritmlar rasmiy ravishda bajariladi, bu degani bajaruvchi bajarilayotgan amallarni mazmunini anglash shart emas. Algoritm tuzish jarayoniga algoritmlashtirish deyiladi.

Algoritm tuzish jarayonida nazariy va amaliy nuqtai nazardan algoritmlash,

dasturlash va EHM larni qo’llash bilan bog’liq bo’lgan bilimlar kerak. Asosiy maqsad bu masalani qo’yish, masalaning yechish algoritmini tuzish, algoritmi mashina dasturi ko’rinishida amalga oshirish va algoritmni samaradorligini ko’rsatish muammolarini o’rganish. Bu jarayonlar algoritmni to’liq yaratish tushunchasiga olib keladi va quyidagi bosqichlarni belgilaydi:

1. Masalaning qo’yilishi
2. Modelini yaratish
3. Algoritmni ishlab chiqish
4. Algoritm to’g’riligini tekshirish
5. Algoritmni amalga oshirish
6. Algoritmni va ularning murakkabligini tahlil qilish.
7. Dasturni tekshirish
8. Hujjatlashtirish

Masalaning qo’yilishi

Masalani yechishdan oldin, uni berilishini aniq shakllantirib olish zarur. Bu jarayon to’g’ri savollarni aniqlash bo’lib, savollar quyidagicha bo’lishi mumkin:

1.1. Dastlabki berilgan masala shartlarida hamma iboralar tushunarlimi?

1.2. Nima berilgan?

1.3. Nimani topish kerak?

1.4. Yechimni qanday ta’riflash kerak?

1.5. Qaysi berilganlar yetarli emas va hammasi kerakmi?

1.6. Qanaqa mumkinliklar qabul qilingan?

Albatta, bulardan tashqari boshqa savollarni ham ishlatish mumkin, yoki ayrim savollarni bir necha bor takror ishlatishga to’g’ri keladi

Modelni yaratish

Akademik A.N.Tixanov fikri bo'yicha matematik modellashtirish dunyoni bilish va o'rganisha kuchli qurollardan (vositalardan) biridir. Uning ta'rifi bo'yicha matematik model tashqi dunyoning xodisalar turkumini matematik belgilar yordamida taxminiy tavsifi.

Xodisani tavsiflash uchun uning muhim hususiyatlarini, qonuniyliklarini, ichki aloqalarini, ayrim xossalarning ahamiyatini aniqlash zarur. Eng muhim faktorlari aniqlanganda, ahamiyatlari kamroq bo'lganlarini hisobdan chiqarish mumkin. Umuman, modelni tanlash fandan ko'ra ko'proq san'at ishi deb hisoblanadi, yahshi tuzilgan modellarni o'rganish esa – modellashtirishda tajriva orttirishning eng yaxshi usuli. Modelni yaratishda quyidagi savollarni aniqlash maqsadga muvofiq:

2.1. Masalani yechish uchun qaysi matematik struktura ko'proq mos keladi?

2.2. O'xshash masalaning yechimi bormi?

2.3. Masalaning barcha muhim ma'lumotlari matematik ob'yektlar orqali tavsiflanadimi?

2.4. Izlanayotgan natija biron bir matematik o'lchamga mos keladimi?

2.5. Modelning ob'yektlari orasidagi bog'lanishlar aniqlanganmi?

2.6. Tuzilgan model bilan ishlash qulaymi?

Algoritmni ishlab chiqish

Algoritmlashtirish jarayoni uslublari bo'yicha matematik modellarni tuzish jarayoniga juda yaqin. Har bir algoritmni ishlab chiqish bevosita o'ziga xos yondashishni talab qilishiga qaramasdan, bu faoliyatni umumiy uslub va bosqishlari ham mavjud. Ba'zan dasturlarni tezroq yozib boshlashga hohish paydo bo'ladi. Lekin bu xatoli, chunki aynan algoritmni ishlab chiqish bosqichiga

va uning to’g’riligiga masalaning to’liq yechimi bog’liqdir. Algoritmlarni tuzish turli xil uslublari mavjud.

Algoritmni to’g’riligini tekshirish

Dastur to’g’riligini isbotlashning eng keng tarqalgan turi – bu uni testlardan o’tkazishdir. Algoritmni tekshirishda nazoratchi boshlang’ich ma’lumotlarni majmui algoritmik test deb nomlanadi.

To’g’ri deb shunday algoritmga aytiladiki, u masalaning qo’yilishida talab qilinadigan natijani har qanday ruxsat etilgan boshlang’ich ma’lumotlar bilan ham shakllantirib biladi. Odatda, dastur bergan natijalar ma’lum bo’lgan yoki qo’lda hisoblangan ma’lumotlar bilan taqqoslanadi, va ular to’g’riligi aniqlansa dastur to’g’ri ishlaydi degan hulosaga kelish mumkin. Ammo bu usul bilan foydalanuvchini hamma shubhalardan xalos qilib bo’lmaydi, ya’ni dastur ishlamaydigan hamma holatlarni hisobga olib bo’lmaydi.
Gudman va Xidetniyemi lar tomonidan algoritm to’g’riligini isbotlash uchun quyidagi uslubiyat taklif qilingan.

Algoritm 0 dan **m** gacha bo’lgan qadamlar ketma-ketligi ko’rinishida tavsiflangan deb tahmin qilaylik. Har bir qadam uchun qandaydir asoslanishni taklif etamiz. Xususan, qadamdan oldin va keyin ishlaydigan shartlar haqida lemma kerak bo’lishi mumkin. Shu bilan birgalikda, algoritm chekliligining isbotini ham taklif etamiz, va hamma ruxsat etilgan kiritish ma’lumotlarini tekshirib, hamma mumkin bo’lgan chiqarish ma’lumotlarni olamiz. Algoritmni to’g’riligi bilan samaradorligi o’rtasida hech qanday aloqa yo’qligini ta’kidlab o’tamiz. Aslida hamma talablarga bir xil yahshi javob beradigan algoritm kamdan-kam ishlab chiqiladi.

Algoritmni amalga oshirish

Algoritmni amalga oshirish deganda, EHM uchun dasturni yozish deb tushuniladi. Buning uchun quyidagi savollarga javob berish kerak:

5.1. Asosiy o’zgaruvchilarni aniqlash.

5.2. O’zgaruvchilarning turlarini aniqlash.

5.3. Nechta massiv yoki fayllar va qanday kattalikda ular kerak bo’ladi?

5.4. Bog’lanilgan ro’yxatlardan foydalanish ma’nolimi?

5.5. Qanday dasturiy qismlar kerak bo’lishi mumkin (tayyor bo’lsa ham)?

5.6. Qaysi dasturlash tilini tanlash ?

Dastur yozish yoki tuzishning hilma-hil usullari va uslublari mavjud.

Algoritmni va uning murakkabligini tahlil qilish

Algoritmni tahlil qilishdan maqsad - algoritmga ma’lumotlarni aniq muvaffaqiyatli qayta ishlash uchun kerak bo’ladigan xotira hajmi va ishlash vaqtining baholari va chegaralarini olish. Bir masalani yechadigan ikki algoritmni taqqoslash uchun qandaydir sonli mezon topish kerak.

Faraz qilaylik, A - qandaydir bir turkumdagi masalalarni yechadigan algoritm, n - esa shu turkumdagi alohida bir masalaning kattaligi. Umumiy holda, n - oddiy skalyar yoki massiv yoki kiritiladigan ketma - ketlikning uzunligi bo’lishi mumkin. $f_A(n)$ - n kattalikdagi ixtiyoriy masalani yechadigan algoritm A bajarish kerak bo’lgan asosiy amallarni (qo’shish, ayirish, taqqoslash,...) yuqori chegarasini beradigan ishchi funksiya.

Dasturni tekshirish

Biz dasturni har bir qismini tekshiradigan kirituvchi ma’lumotlar to’plamini tanlashimiz kerak. Ko’p murakkab algoritmlarni matematik tomondan tadqiq qilish yoki juda qiyin yoki mumkin emas. Bunday holatlarda algoritmni faoliyat jarayonida va qiyinligi bo’yicha tekshiradi. Bundan tashqari dasturlami hisoblash imkoniyatlarini aniqlash uchun ham testlash maqsadga muvofiq. Ko’p dasturlar qandaydir kiritiladigan ma’lumotlar bilan yahshi ishlasa, boshqalari bilan yomon ishlaydi. “Yahshi” lardan “yomon” larga o’tish “mayin” bo’lish kerak. Testlash uchun ma’lumotlar dasturning qiyinligiga, mavjud vaqt resurslariga, kiritish-

chiqarishsoniga bog’liq holda tanlanadi. Bu yerda analitik va eksperimental tahlil bir- birini to’ldiradi.

Hujjatlashtirish

O’zingiz yozmagan dastur kodini o’qish juda qiyin. Bu muammoni hujjatlashtirish yordamida yechsa bo’ladi. Hujjatlashtirish o’z ichiga hamma yordamchi ma’lumotlarni oladi va dasturda nima bajarilishini tushuntirib beradi, xususan, blok-sxemalardagi boshqarishni uzatish, berilganlarni kiritish-chiqarish shaklini batafsil tavsif qilish, siklning parametrlari, yordamchi local va global proseduralarni bajarilishi va boshqalar.

Hujjatlashtirishning eng asosiy qoidasi bu “boshqalar yozgan dasturlarni qanday ko’rishni istasangiz, o’zingiz ham dasturni shunday ko’rinishda rasmiylashtiring”. [18]

1.3 Masalaning yechish dasturini ishlab chiqish

Masala: Evklid algoritmi

Masalaning qo’yilishi:

Ikkita butun musbat m va n sonlar berilgan. Ularning eng katta umumiy bo’luvchisi (EKUB)ni topish talab qilinadi. Ya’ni eng katta butun musbat son topish kerakki, uni M va N ga bo’lganda butun son chiqsin.

1-usul:

Algoritmni tuzish:

1. Boshlash;
2. M ni N ga bo’lamiz, qoldiq r ga teng bo’lsin;
3. Agar r=0 unda n natija; 5 o’ting;
4. M:=N; N:=r; 2 o’ting;
5. Tamom.

Algoritm tahlili

Shu algoritmni tadqiq qilib ko'raylik. m=119, n=544 deb qabul qilaylik. Ikkinchi qadamdan boshlaymiz. Algoritmga binoan bo'lish natijasini nolga teng deb hisoblaymiz va r ga 119 ni ta'minlaymiz, keyin 3-qadamga o'tamiz. **R** nolga teng bo'lmaganligi uchun, hech nima qilmaymiz va 4-qadamga o'tamiz. Bu yerda m ga 544 ni, n ga 119 ni ta'minlaymiz. Umuman, ravshan bo'ldiki, m<n bo'lsa, 2-qadamda m va n larga nisbatan hech qanday amallar bajarilmaydi, algoritm esa m va n o'zgaruvchilar qiymatlari o'rin almashishiga olib keladi.

Algoritm optimallashtirish

Algoritmni optimallashtirish uchun unga quyidagi qadamni qo'shamiz:

1.2. agar m<n t:=n; n:=m; m:=t;

Endi 2-qadamga kelsak, 544:119=4,68/119. r ga 68 ni ta'minlaymiz. 3-qadam ishlamaydi. 4-qadamda n=68, m=544, r=68. Keyingi sikllarda (r=51, m=68, n=51), keyin (r=17, m=51, n=17) va 51/17, ya'ni r=0.

Shunday qilib, algoritm sikli 3- qadamda tugadi va 544 va 119 laming umumiy bo'luvchisi 17 ga teng bo'ldi.

Bu algoritm umumiy bo'luvchini topish uchun yagona emas. Bunday algoritmni topish uchun Dj. Steynning binar algoritmi, yoki V. Xorrisning algoritmidan foydalaniladi.

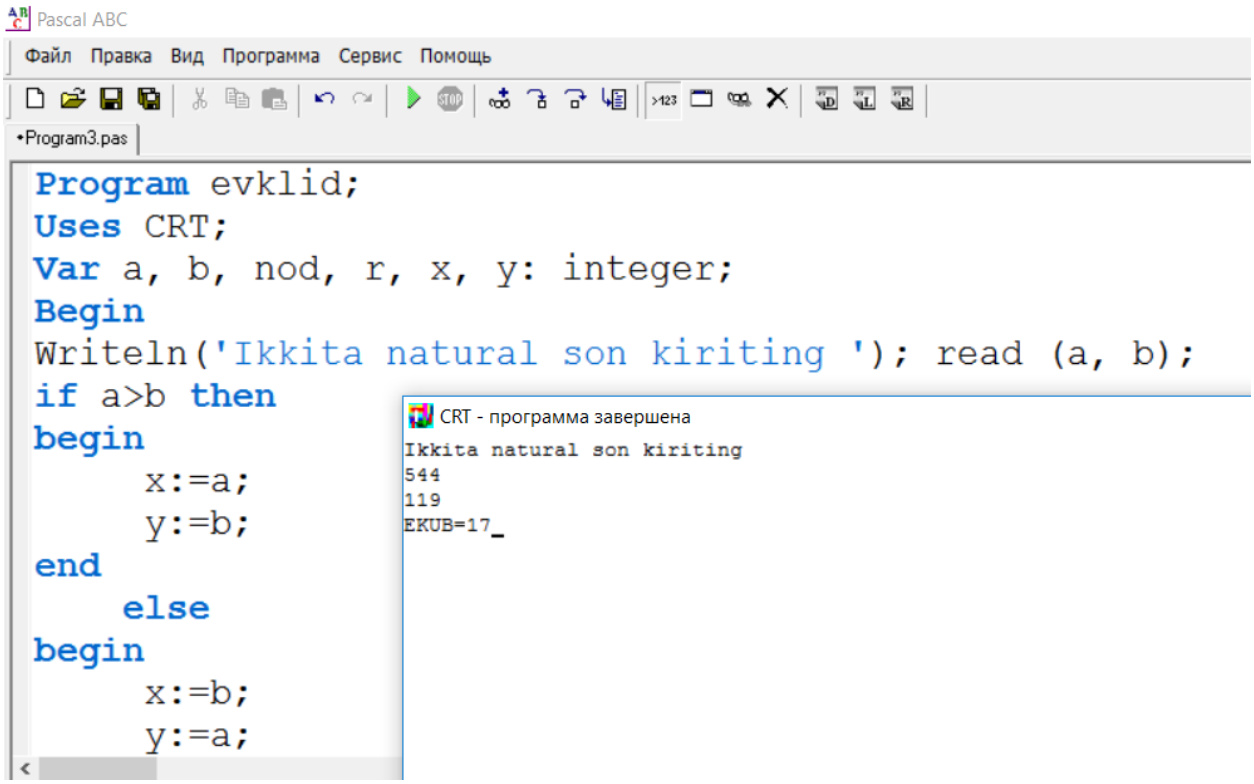
Algoritmni amalga oshirish

Shu algoritmni kompyuterda amalga oshirish uchun quyidagi Paskal dasturida tuzilgan dastur kodini keltirish mumkin:

```
Program evklid;
Uses CRT;
Var a, b, nod, r, x, y: integer;
Begin
Write('Ikkita natural son kiriting '); read (a, b);
```

```
if a>b then
        begin
            x:=a;
            y:=b;
        end
   else
        begin
            x:=b;
            y:=a;
        end;
if (x>0) and (y>=0) then
begin while y<>0 do
        begin
            r:=x mod y;
            x:=y;
            y:=r;
        end;
nod:=x; write (’EKUB=’,nod);  end
        else write (‘berilganlarda xato’);
end.     [18]
```

Dastur kodini Pascal ABC dasturiga kiritamiz va kompilyatsiya qilamiz, va natija quyidagicha.



2-usul ***Algoritmni tuzish:***

Boshlansin

A, B kiritilsin

Toki A>B

Agar A>B

U holda

A:=A-B

Aks holda

B:=B-A

Oxiri

Oxiri

A, B chiqarilsin

Tamomlansin.

Evklid algoritmi yordamida bir nechta natija oling.

N=15 va M=25 bo’lsa:

(15, 25) (15, 25-15) -» (15, 10) (15-10, 10) -» (5, 10) -» (5, 10-5) (5, 5), javob 5.

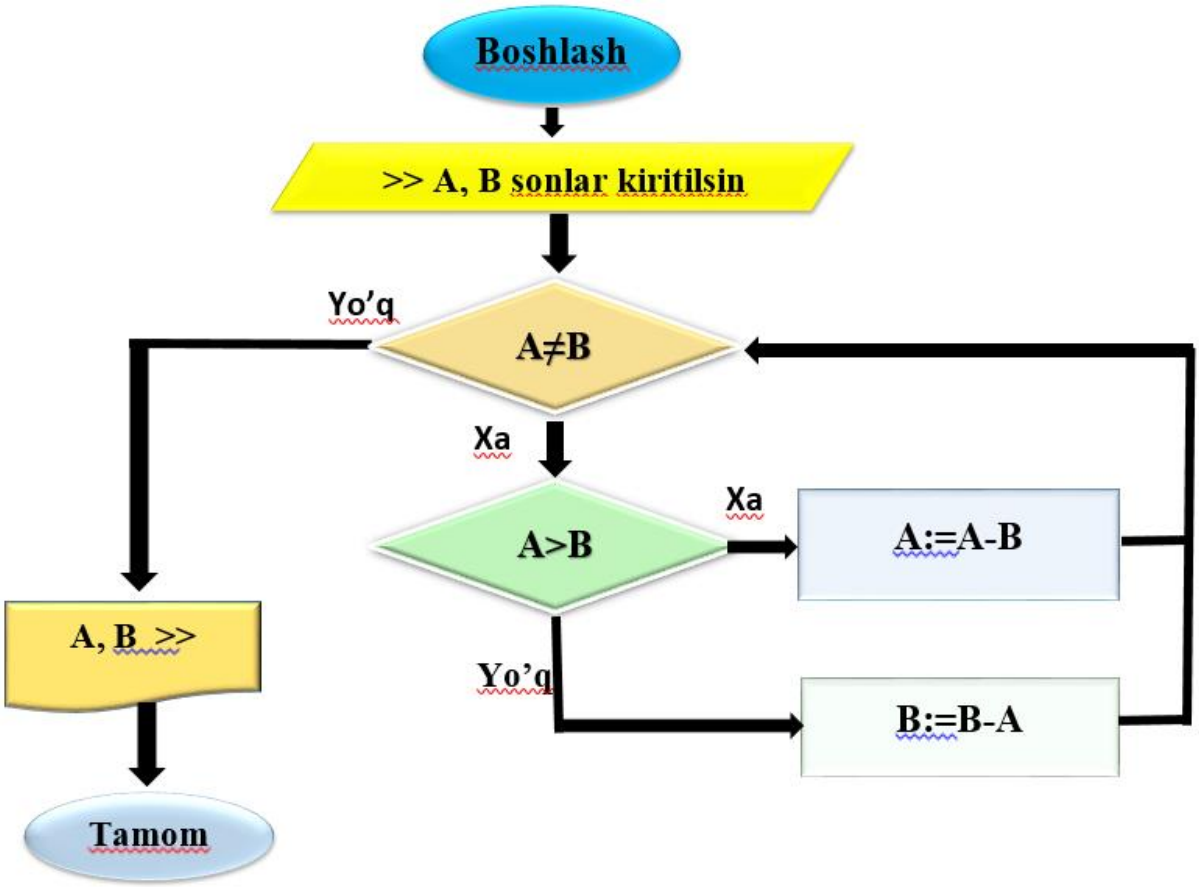
N=18 va M=39 bo’lsa:

(18, 39) (18, 39-18) -» (18, 21) -» (18, 21-18) -» (18, 3) (18-3, 3) (15, 3) -» (15-3, 3) -» (12, 3) (12-3, 3) (9, 3) -» (6, 3) (6-3, 3) -» (3, 3), javob 3.

N=91 va M=65 bo’lsa:

(91, 65) -» (91-65, 65) (26, 65) -» (26, 65-26) (26, 39) -» (26, 39-26) -» (26, 13) -» (26-13, 13) -» (13, 13), javob 13.

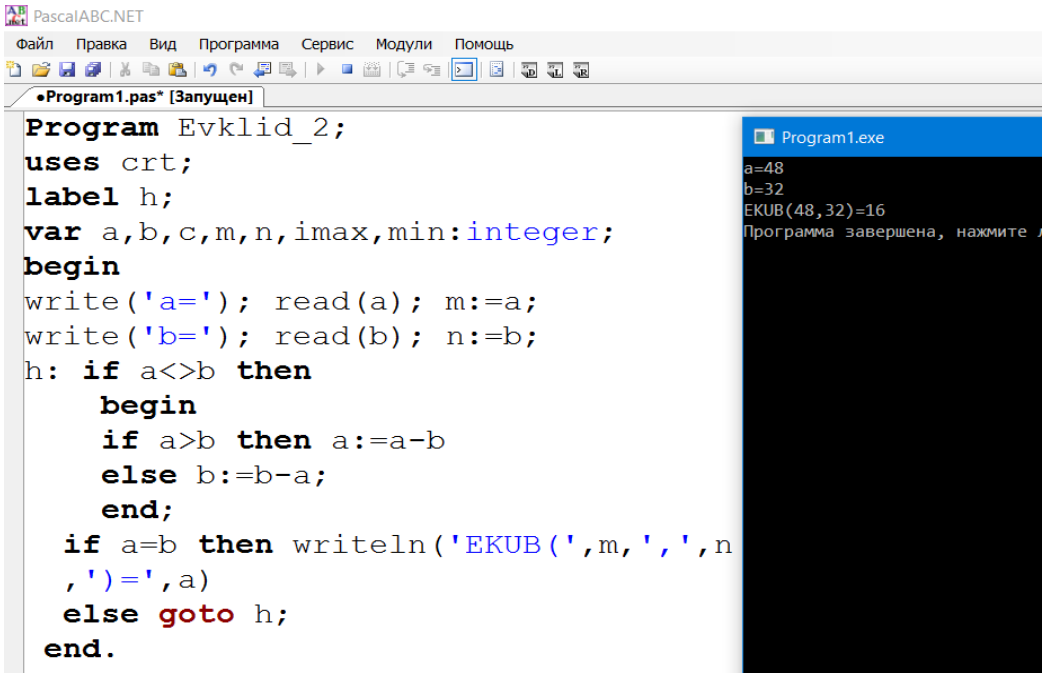
Evklid algoritmining blok-sxema orqali ifodalanishi



EKUB ni topuvchi algoritmni Paskal dasturlash tilida yozamiz:

```
Program Evklid_2;
uses crt;
label h;
var a,b,c,m,n,imax,min:integer;
begin
write('a='); read(a); m:=a;
write('b='); read(b); n:=b;
h: if a<>b then
    begin
    if a>b then a:=a-b
    else b:=b-a;
    end;
  if a=b then writeln('EKUB(',m,',',n,')=',a)
   else goto h;
 end.
```

Dastur kodini Pascal ABC.NET dasturiga kiritamiz va kompilyatsiya qilamiz. Natija quyidagicha



EKUB ni topuvchi dasturlarni taqqoslash

1-usulda ikkita kiritilgan sondan kattasi aniqlab olinib, katta sonni kichigiga bo'lib qoldiq qismini r ga o'zlashtirib olayabdi, va katta songa kichik sonni o'zlashtirib, kichik songa esa qoldiqni o'zlashtirib, qoldiq r=0 bo'lgunga qadar davom ettirayabdi. r=0 bo'lganda x ning qiymatini umumiy bo'luvchi deb natija sifatida chop etayapdi.

2-usulda ham ikkita sondan kattasi aniqlab olinib, katta sondan kichigini ayirib, ikkala son bir-biriga teng bo'lgunga qadar davom ettirilayabdi. Qachon ikkala son ham bir-biriga teng bo'lsa shu sonni umumiy bo'luvchi deb topib, chop etayabdi.

Ikkala usuldan 1-usulda natija chiqishi uchun bajariladigan qadamlar soni kamroq ekan. 1-usul optimal yechim.

For tipidagi sikl yordamida bajariladigan algoritm va dastur tuzish

For (uchun) tipidagi sikl oldindan takrorlanishlar soni ma'lum bo'lganda ishlatiladi

2.1- Misol: Sonli massiv A=(a_1,a_2,…,a_N) ning elementlarini yig'indisini hisoblang. Test

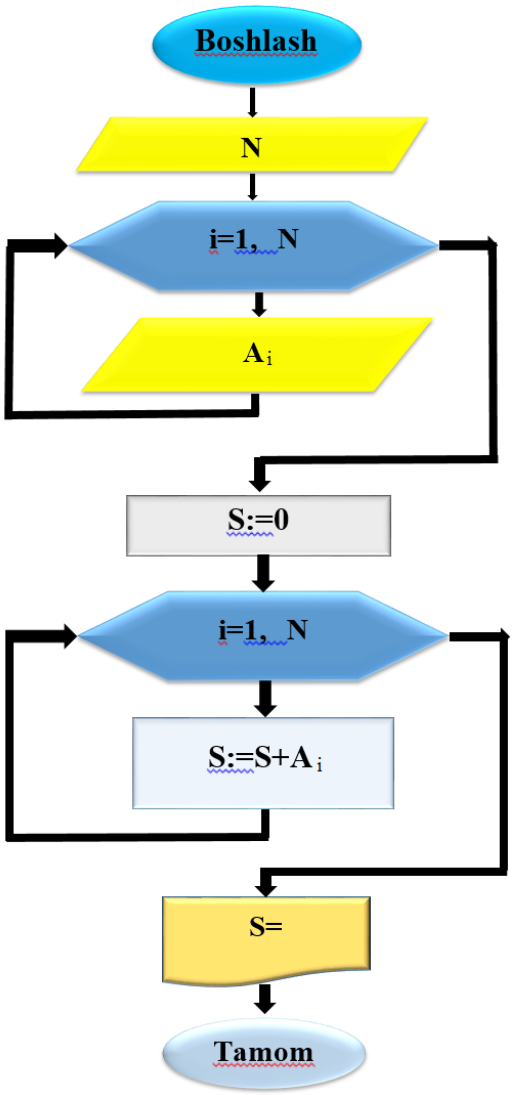
Berilgan		Natija
N=5	A=(3, 5, -2, 6, 3)	S=15.0
N=4	A=(4, 7, 2, 5)	S=18.0

Algoritmi:

```
alg Summa (but N,
           haq jad A[1:N], haq S)
arg N,A
boshl but i
  S:=0
  sb i uchun 1 dan N gacha
     S := S + A[i]
  so
tamom
```

Algoritmning bajarilishi

i	S
	0
1	$0 + a_1 = 0+3 = 5$
2	$a_1 + a_2 = 3+5 = 8$
3	$a_1+a_2+a_3 = 8-2 = 6$
4	$a_1+a_2+a_3+a_4 = 6+6 = 12$
5	$a_1+a_2+a_3+a_4+a_5 = 12+3=15$



Turbo Pascaldagi dasturi:

```
Program Summa;
Type Mas = Array [1..20] of Real;
Var A : Mas;
i, N : Integer;
S : Real;
BEGIN
ReadLn(N);
For i := 1 to N do
begin
Write('A [ ', i , ' ] = ');
ReadLn(A[i])
end;
S := 0;
For i := 1 to N do S := S+A[i];
WriteLn( S : 5 : 1);
ReadLn;
End.
```

[18]

II. BOB Dasturlash uslubiyatlari va masalani yechish metodlari

2.1 Dasturlash uslubiyati.

Masaladan dasturga

Endi shu paytgacha ko’rib chiqilgan dasturlash vositalarini ko’plab qiziqarli dasturlar yaratishga qo’llaymiz. Agar dasturchi oldiga murakkab masala qo’yilgan bo’lsa, unga mos dasturni yaratish uchun uzoq vaqt va ko’p harakat talab qilanadi. Shuning uchun dasturni yozishda ma’lum qoidalar to’plamiga yoki dasturlash uslubiga rioya qilish kerak bo’ladi.

Masala dasturini yaratish jarayonini quyidagi bosqichlarga bo’lish mumkin:

1. Masalani ta’riflash.

2. Yechish usuli (algoritm)ni tanlash va tasvirlash.

3. Dasturni yaratish.

4. Dasturni tekshirish.

5. Dasturni takomillashtirish.

6. Dasturni tuzatish.

Quyida, sanab o’tilgan qismlarning har birini alohida ko’rib chiqamiz.

Masalani ta’riflash.

Dasturni yaratishda, eng avval berilgan masalaning maqsadi va bo’lajak dasturga qo’yiladigan talablarni bilish kerak. Aks holda, dasturchi ayrim tafsilotlarni nazarga olmasligi va masalani yechish uchun zarur bo’lgan, ammo uning tasvirida aniq ifodalanmagan amallar dasturda o’z aksini topmasligi mumkin. Masalani ta’riflashda dasturda qaysi dastlabki ma’lumotlardan foydalanish qanday natijalar olish kerakligi aniq ko’rsatilishi kerak. Talab qilinayotgan natijalarni olish uchun zarur bo’lgan amallarga kelsak, ular keyinroq, dastur tuzishda aniqlanadi.

Misolni ko’rib chiqamiz. Yillarni ifodalovchi dastlabki ma’lumotlar ichidan kabisa yillarini aniqlaydigan dastur tuzish kerak. Shubxasiz, bunday topshiriqdan masalani aniq ta’riflash mumkin emas.

Birinchidan, dastlabki ma’lumotlarning o’zgarish chegaralari ko’rsatilmagan. Chunki yuz yillikni bildiradigan kabisa yillari alohida formula bilan aniqlanadi, shuning uchun dastlabki ma’llumotlar bizning eramizning yuz yilliklarini yoki eramizgacha bo’lgan yillarni bildiradimi shuni bilish zarur. Ikkinchidan, EHM natijalarini qaysi ko’rinishda bosib chiqarishi kerakligi aytilmagan.

Masalaning ta’rifini aniqlaymiz. Dastlabki ma’lumotlar – 1 dan 2000 gacha bo’lgan bizning eramizning yillarini ko’rsatuvchi sonlar. Shunday dastur yozish kerakki, unga asosan EHM, agar berilgan yil kabisa yili bo’lsa, KABISA va aks xolda ODDIY so’zlarini bosib chiqarsin. Masalani bunday ta’riflash ancha aniq.

Boshqa misolni ko’rib chiqamiz. Berilgan sonlar yig’indisini hisoblash kerak, bu yerda dastlabki ma’lumotlar sonini yoki sonlarning berilgan ketma-ketligi uzunligini aniqlash usulini bilish kerak. Bunday masalani ta’riflashning ayrim variantlarini keltiramiz:

1. dastlabki ma’lumotlar – ikkita butun son. Ularni dasturga kiritish, yig’indisini hisoblash va bu yig’indini bosib chiqarish kerak.

2. dastlabki ma’lumotlar – o’nta butun sonlar. Ularni dasturga kiritish, yig’indisini hisoblash va bu yig’indisini bosib chiqarish kerak.

3. dastlabki ma’lumotlar – 0 ga teng bo’lmagan sonlar ketma-ketligi. Ketma-ketlikning oxiri – nol. Dastlabki ma’lumotlarni o’qish, yig’indisini topish va uni bosib chiqarish kerak.

Yechish usuli (algoritmi)ni tanlash va tasvirlash.

Masalani yechish, EHM dasturda aniqlanadigan katta hajmdagi hisoblash ishlarini bajaradi. Masalani yechuvchi dasturchi esa yechish usulini, ya’ni

qo’yilgan masalaning algoritmini bilish zarur. Masalan, kabisa yillarini aniqlovchi dastur tuzishdan avval, dasturchi kabisa yilini aniqlash qoidasini bilish kerak.

Agar masalani yechishning bir neha usullari ma’lum bo’lsa uni tanlashda usulning universalligi, soddaligi va tejamliligini hisobga olish kerak. Tanlanmagan usul dasturning maksimal tezkorligini, xotiraning eng kam sarflanishi va dasturlash nuqtai nazaridan eng yaxshi hususiyatlarini ta’minlashi kerak.

Dasturni yaratish.

Tajriba shuni ko’rsatadiki, har bir dastur faqat o’ziga hos hususiyatlarga ega, shu bilan birga yangi yaratilayotgan dasturni bir nechta, avval yozilgan, alohida dasturlardan yaratish ham mumkin. Dasturlash - bu ijodiy jarayon, shuning uchun dastur tuzishning umumiy usuli yo’q. Ammo ba’zi usullarni tavsiya qilish mumkin, ulardan foydalanib, dasturchi o’z ishini yengillashtiradi va uni ancha oson bajaradi. Bunday usullardan biri – masalani qismlarga ajratish. Katta masalalarni dasturlashda ma’lum qiyinchiliklar uchraydiki, amalda ularni ta’riflash mumkin emas. Bunday hollarda masala, bir-biriga bog’liq bo’lmagan, dasturlanadigan qismlarga ajratiladi. Agar alohida qism hali ham ancha katta bo’lsa, uni qismlarga ajratishni, masalaning hosil qilingan qismlari ancha sodda va aniq bo’lgunga qadar, davom etishi kerak. Bu usulni ta’riflangan masala misolida tushuntirib beramiz.

Dastlabki ma’lumotlar – bizning eramizning 1-dan 2000-gacha bo’lgan yillariga mos keluvchi sonlar. Shunday dastur yozish kerakki, u bajarilganda EHM kabisa yili uchun KABISA, aks holda ODDIY so’zlarini bosib chiqarsin. Bu masalani qismlarga ajratamiz:

1. Dastlabki ma’lumotlarni kiritish.

2. Berilgan yil kabisa yil ekanini tekshirishi.

3. Natijalarni bosib chiqarish.

Ma’lumotlarni kiritish va natijalarni bosib chiqarish bu yerda ortiqcha qiyinchilik tug’dirmaydi, chunki ularga Turbo Paskal tilining ma’lum operatorlari

mos keladi. Agar kabisa yillarini aniqlaydigan funksiya ma’lum bo’lsa, dasturning ikkinchi qismini amalga oshirish osonlashadi. Bunday funksiyani keyinroq yozamiz, hozir esa yaratiladigan dasturning loyihasi (chizmasi)ni tuzamiz.

```
Program kabisalar;

Var yil:integer {(yil>=1) and (yil<=2000)};

Function kabisa (y:integer):Boolean;
```

Agar y yil kabisa bo’lsa, funksiyaning qiymati True, aks holda – false bo’ladi.

```
Begin
    Read(yil);
    If kabisa (yil) then write(’KABISA’)
    Else write(’ODDIY’)
End.
```

Kabisalar dasturida kabisa funksiya tasviri to’g’ri burchakli to’rtburchak bilan almashtirilgan. Masalaning birinchi va uchinchi qismi dasturlashtirildi, qolgan qismiga esa yangi masala deb qarash mumkin. uning ta’rifa to’g’ri burchakli to’rtburchak ichida yozilgan.

Kabisa funksiyani tasvirlash uchun yechish usuli – qoidasini aniqlab olish zarur. Bu usul (qoida)ga asosan kabisa yillari aniqlanadi:

1. Asrlarni (misol uchun, 1800, 1900, 2000) bildiradigan yillar soni to’rtga bo’linsa, ular kabisa yillari bo’ladi. Masalan, 2000-yil kabisa yil, chunki 18 va 19 sonlari 4 ga bo’linmaydi.
2. 4 ga bo’linadigan qolgan hamma yillar kabisa yillari bo’ladi. Masalan, 1988-yil kabisa, 1987-yil kabisa emas.

Bu qoidalar yordamida funksiyani tuzamiz:

```
Function kabisa (y:integer):Boolean;
Begin
        If y mod 100=0
then {asr}
if  y  mod 400 = 0
then kabisa:=true
else kabisa:=false
else {asr emas}
if  y  mod 4=0 then kabisa:=true
else kabisa:=false
end.
```

Funksiyani dasturdagi to’rtburchak o’rniga qo’yamiz:

```
Program kabisalar;
 Var  yil:integer((yil>=1) and (yil<=2000));
Function kabisa (y:integer):Boolean;
Begin
If  y mod 100 = 0 then  if  y mod 400 =0 then kabisa:=true
Else kabisa:=false
Else {asr emas}
If  y mod 4=0 then kabisa:=true else kabisa:=false;
```

```
End;

Begin {dastur boshi}

Read(yil);

If kabisa yil then write('KABISA') else write('ODDIY')

End.
```

Dasturning mustaqil qismlarini funksiyalar va protseduralar ko'rinishida yozish qulayligi ko'rinib turibdi. Katta dasturlarni yaratishda, odatda, bu usullardan foydalaniladi.

Ko'rib chiqilgan masala anchagina sodda. Uni qismlarga ajratmasdan ham yechish mumkin edi. Biroq, kelgusida bizga murakkab masalalarni yechishga to'g'ri keladi, shuning uchun yuqorida tasvirlangan usulni puxta o'rganib olish muhim.

Yana bir bor takidlaymiz, agar masalani qismlarga bo'lishda hosil bo'lgan qism hali ham juda murakkab bo'lsa, tavsiya qilingan usulni qayta qo'llash kerak. Masalaning bunday qismi so'z bilan ta'riflanadi yoki to'rtburchakda tasvirlanadi. Dasturda birorta ham bunday to'rtburchaklar qolmagandan keyingina dastur tamom bo'ldi, deb hisoblanadi.

Dasturning alohida qismlarini faqat funksiyalar va protseduralar ko'rinishida dasturlash har doim ham foydali bo'lavermasligini misolda ko'rsatamiz. Dasturning yuqorida ko'rgan loyihasini quyidagi ko'rinishda yozish mumkin:

```
Program kabisalar(input, output);

Var yil:integer((yil>=1) and (yil<=2000));

B:Boolean;

Begin

Read(yil);
```

Yilning kabisa ekanligini tekshirish, ha bo’lsa, b o’zgaruvchiga true qiymat, aks holda, false qiymat o’zlashtiriladi.

```
If  b then write(’KABISA’)

Else write(’ODDIY’)

End.
```

To’rtburchakda tasvirlangan dasturning qismini shartli operator yordamida bajaramiz:

```
If  yil mod 100=0 then

If  yil mod 400 =0 then b:=true

Else b:=false

Else  if yil mod 4=0 then b:=true

Else b:=false;
```

Shartli operatorni to’rtburchak o’rniga yozib, tugallangan dasturni olamiz. Muhokama qilinayotgan masalani amalga oshirishning bu ikki ko’rinishini solishtirib, funksiyani o’z ichiga olgan dasturning o’lchami birmuncha katta ekanligini ko’ramiz. Ammo funksiya bu dasturning mustaqil qismi, demak, unda dasturning boshqa qismlarida o’zgaruvchilar qanday nomlanganligini hisobga olmasdan, o’zgaruvchilar uchun boshqa nomlarni ishlatish mumkin.

Dasturni tekshirish

Masalani dasturlashda osongina adashish mumkin.

Dasturning to’g’riligini har xil yo’llar bilan tekshirish mumkin. ulardan eng soddasini ko’rib chiqamiz. U shundan iboratki, aniq dastlabki ma’lumotlarga ega bo’lgan holda dasturchining o’zi, EHM ishtirokisiz, dasturda yozilgan amallarni bajaradi. Bu holda barcha amallarni, xuddi mashina bajaradigandek, ularning

mohiyatini o'ylab ko’rmasdan, beixtiyor bajarishi kerak. Olingan natijalarga qarab, dasturning to’g’riligini muhokama qilish mumkin.

Ikki sonini n-darajaga ko’taradigan dastur yozamiz:

```
Program daraja (input, output);

Var k, n, p:integer;

Begin

Read(n);

P:=1;

For k:=1 to n do

P:=p*2;

Write(p);

End.
```

Bu dasturni tekshirish uchun bir varaq qog’oz olib, unga dastur matnida uchraydigan barcha o’zgaruvchilarni yozib chiqamiz (jadvalning birinchi yo’li:)

O’zgaruvchilar qiymatlari	K	N	P
Dasturning bajarilishidan oldin	?	?	?
Read (n) operatori bajarilgandan keyin	?	3	?
P:=1 operatori bajarilgandan keyin	?	3	1
1-sikl bajarilgandan keyin	1	3	2
2-sikl bajarilgandan keyin	2	3	4
3-sikl bajarilgandan keyin	3	3	8
Dastur bajarilgandan keyin	3	3	8

Dastur bajarilgunga qadar o’zgaruvchilarning qiymatlari aniqlangan emas. Dasturning bajarilishida o’zgaruvchilarga aniq qiymatlar o’zlashtiriladi, ularni jadvalga ustun bo’ylab pastga tomon yozib boramiz.

Daraja dasturda read(n) operator xotiradan n ning qiymatini o’qiydi. N=3, deb faraz qilamiz, p – o’zgaruvchiga 1 qiymat o’zlashtiriladi. Sikl operatorlari bajarilganda k va p o’zgaruvchilarning qiymatlari doim o’zgarib boradi. Hisoblash oxirida k=3 , p=8 bo’ladi. Demak, daraja dastur 2 sonini 3=darajaga to’g’ri ko’taradi, chunki 2^3=8.

Dasturni tashkil qiluvchi funksiyalar va protseduralarni bir-biridan mustaqil ravishda tekshirish olingan natijalarni esa masalaning hammasini tekshirishda ishlatish maqsadga muvofiqdir. Rekursiv protsedura funksiyalarning to’g’riligini aniqlash ancha murakkab. O’zgaruvchilarning qiymatlarini aniq ko’rsatish uchun rekursiv protsedura yoki funksiyaning har bir nusxasini alohida jadvalga yozish kerak.

Dasturning to’g’riligini tekshirishda foydalaniladigan ma’lumotlar nazorat ma’lumotlari, deyiladi. Qulay va shu bilan birga aynan shu masala uchungina xos nazorat ma’lumotlarini tanlash juda muhimdir. Masalan, agar dasturga sikl kiritilgan va uning takrorlanish soni dastlabki ma’lumotga bog’liq bo’lsa, bu holda, nazorat qiymat sifatida shunday qiymatni tanlash maqsadga muvofiqki, unda sikl amallarini ko’p marta bajarish talab qilanmaydigan bo’lsinl aks holda siklni tekshirish ko’p vaqtni oladi.

Yuqorida yozilgan daraja dasturini tekshiramiz. N=3 da dastur to’g’ri tuzilganligi ko’rsatilgan edi. Yana n=0 va n=1 bo’lgan ikki holni ko’ramiz. Bu qiymatlar bilan dastur amallarini bajarib, mos ravishda p=1 va p=2 natijalarini olamiz. N=0 da dastur sikli bajarilmaydi va p o’zgaruvchining qiymati 1 ga teng bo’lib qolaveradi. Demak, daraja dastur, 2 sonini, n>=0 darajaga ko’tarishga mo’ljallangan. Shunday qilib, siklning to’g’riligini tekshirish uchun shunday nazorat qiymatlarini tanlash kerakki, unda sikl bir marta, bir necha marta

bajariladigan yoki umuman bajarilmaydigan bo'lsin. Shartli operatorni tekshirishda uning har bir qismi (tarmog'i) hech bo'lmaganda bir marta bajariladigan bo'lishi kerak. Masalan, kabisa funksiyaning to'g'riligiga ishonch hosil qilish uchun ushbu 1600, 1900, 1988, 1987 nazorat ma'lumotlardan foydalanamiz. Funksiyada ko'rsatilgan amallarni bajarib, true, false to'g'ri natijalarini olamiz.

Ba'zi nazorat qiymatlarda dastur natijalari to'g'ri bo'lsa, dasturning o'zi ham to'g'ri tuzilganligini tasdiqlash mumkinmi? Yo'q albatta!

Boshqa dastur misolida bu fikrning to'g'riligini tushuntirib boramiz. 2 sonini n darajaga ko'tarish uchun quyidagi dastur tuzilgan bo'lsin:

```
Program daraja (input, output);
Var k, n, p :integer;
Begin
    Read (n) ;    p:=1;
    For k:=1  to n do
    P:=k*2;
    Write (p);
End.
```

Daraja dasturning natijalari 0,1,2 dastlabki qiymatlarda mos ravishda 1, 2, 4 ga teng, ya'ni daraja dasturning natijalari bilan mos keladi. Biroq n>=2 da noto'g'ri natijalar chiqishiga ishonch hosil qilish qiyin emas. Masalan, n=3 qiymatlar p 8 ga teng emas, 6 ga teng bo'ladi.

Dasturning to'g'riligini tekshiradigan boshqa ishonchli usul shundan iboratki, masalaning ta'rifi bo'yicha dastlabki ma'lumotlarning mumkin bo'lgan har qanday qiymatlarida dasturning to'g'riligini isbotlashdir. Dasturning natijalarini aniq qiymatlar bo'yicha tekshirishga qaraganda bu ancha murakkab. Har bir dastur tuzilishining to'g'riligini ko'rsatishga (asosan, misollarda) harakat qilib ko'ramiz.

O’zlashtirish operatori. Operatorning o’ng qismiga o’zgaruvchining qiymatini hisoblaydigan arifmetik ifoda yoziladi. Agar u to’g’ri yozilgan bo’lsa, o’zlashtirish operatori ham to’g’ri bo’ladi. U holda ifodaga tegishli hamma o’zgaruvchilarning qiymatlari aniqlangan yoki EHM xotirasiga kiritilgan yoki protsedura bajarilishida hisoblangan yoki bundan oldingi o’zlashtirish operatorlarga o’zlashtirilgan bo’lishi kerakligiga e’tiborni qaratish zarur. Ifodalarga o’zgaruvchilarning noma’lum qiymatlarini ishlatish dastur tuzishni o’rganayotgan dasturchilarning ko’p takrorlaydigan xatosidir. Barcha aytilganlar shartli operatorlardagi (if dan keyingi) va sikllardagi (while dan keying) mantiqiy ifodalarga ham taaluqlidir. Shartli operator. Misol ko’rib chiqamiz. Dastlabki ma’lumot butun a soni bo’lsin, b natija a sonining mutlaq qiymatiga teng bo’lishi kerak. Bu masalani yechish uchun har xil dasturning to’rt parchasini ko’ramiz.

1. If a < 0 then b:=-a;

a<0 bo’lganda b:=-a operator bajariladi, ya’ni a o’zgaruvchining manfiy qiymatlari uchun shartli operator to’g’ri yozilgan. Agar a>=0 bo’lsa, dastur natijasi aniqlanmagan. Demak, bu dastur noto’g’ri tuzilgan.

2. b:=a;

If a<0 then b:=-a;

a<0 bo’lganda b:=-a operator bajariladi, ya’ni hosil qilingan natija to’g’ri bo’ladi, a>=0 da b=a, chunki shartli operator bajarilmaydi. Bu natija ham to’g’ri. Demak, ikkinchi holda dastur to’g’ri tuzilgan.

3. if a>0

Then b:=a

Else if a<0 then b:=-a;

a>0 bo’lganda b ning qiymati a ga a<0 da esa –a ga teng. Ikkala natija ham to’g’ri , lekin a=0 hol ko’rib chiqilmadi. A=0 da b aniqlangan bo’ladi, chunki shartli operatorning hech qaysi tarmog’i bajarilmaydi. Demak dastur noto’g’ri tuzilgan.

4. if a>0

Then b:=a

Else b:=-a;

A>0 bo’lganda b:=-a operator bajariladi. Qolgan hamma hollarda b:=-a. olingan natijalar to’g’ri. barcha mumkin bo’lgan dastlabki ma’lumotlarning qiymatlarida to’g’ri natijalar hosil bo’layotgani uchun dasturning to’g’riligi haqida hulosa chiqarish mumkin.

Sarlavhada while bo’lgan sikl. Siklni tekshirishda birinchi aniqlanadigan narsa shuki, har qanday mumkin bo’lgan boshlang’ich qiymatlarda bu sikl yakunlovchi bo’ladimi?

Masalan,

```
While a>0 do
Write(’A’);
```

Sikl a>0 da cheksiz bajariladi, chunki bu o’zgaruvchining qiymati siklning har bir takrorlanishida o’zgarmaydi.

Sikl ishining tugallanishini ta’minlovchi zaruriy (ammo yetarli bo’lmagan) shartini ko’rsatamiz. Zaruriy shart – bu sikl sarlavhasidagi shartiga kiruvchi biror o’zgaruvchining qiymati sikl bajarilganda, albatta, o’zgarishi kerakligdir. Misol ko’ramiz:

```
While a>0 do
A:=a+1;
```

Sikl ichida uning o’zgaruvchisining qiymati o’zgarib borishi ko’rinib turibdi, ammo bu sikl tamom bo’lmaydi, chunki a ning qiymati doimo oshib boradi.

Boshqa misol keltiramiz

```
While a>=0 do
A:=a-1;
```

Bu yerda sikl chekli. Siklning har bir bajarilishida a o’zgaruvchining qiymati manfiy bo’lgunga qadar 1 ga kamayib boradi. Unda sikl sarlavhasida yozilgan shart buziladi va, demak, sikl o’z ishini tamomlaydi. Har qanday boshlang’ich qiymatlarda siklning to’g’riligini tekshirish ancha murakkab, chunki buning uchun siklda ko’rsatilgan amallarni ko’p marta bajarish kerak.

Keyinchalik aynan bir masalani tahlil qilishda while siklni tekshirishga yana qaytamiz.

Sarlavhasida For bo’lgan sikl. Bunday turdagi sikllar quyidagi umumiy hususiyatiga ega: agar sikl ichida uning o’zgaruvchisining qiymati o’zgarmasa bu sikl chekli va uning takrorlanish soni sikl sarlavhasida aniqlangan bo’ladi. For siklning bu hususiyatini dasturchi dastur yaratayotganda yoddan chiqmasligi kerak.

Protseduralar va funksiyalarni dasturning mustaqil qismi deb, tekshirish tavsiya qilinadi. Rekursiv funksiyalar va protseduralar haqida alohida fikr yuritamiz. Ular huddi siklga o’xshash cheksiz bajarilishi mumkin. masalan :

```
Procedure p;
Begin
    P; write(’A’);
End.
```

Funksiyani yoki protsedura rekursiv bo’lganda, dastlabki ma’lumotlar (parametrlar) ning har qanday qiymatlarida dastur chekli bo’lishini tekshirish

kerak. Rekursiv funksiyalar va protseduralar chekli bo'lishi uchun ikki zaruriy (biroq, yetarli bo'lmagan) shartiga rioya qilish kerak:

1. Rekursiv funksiya (protsedura) ga murojaat qilish ma'lum. Shartda bajarilishi kerak (masalan, rekursiv murojaat qilishi shartli operatorning qismi bo'ladi) ;

2. Rekursiv funksiya (protsedura) ning tasvirida hech bo'lmaganda bitta o'zgaruvchining qiymati o'zgarishi kerak. Rekursiv murojaatning bajarilishi ana shu o'zgaruvchiga bog'liq bo'ladi.

Sonning faktorialini hisoblovchi funksiyani ko'ramiz. N – dastlabki ma'lumot (n>=0), n! Faktorial – funksiya qiymatidir:

```
Function f ( n:integer): integer;
Begin
    If  n>0   then f:=f(n-1)*n
        Else f:=1;
End.
```

Bu misolda har ikki zaruriy shart bajarilgan. Rekursiv funksiyaga murojaat qilish shartli operatorga kiradi. n – o'zgaruvchining qiymati har bir f(n-1) ga murojaat qilishda o'zgaradi, ya'ni funksiyaning yangi nusxasida parameter qiymatidan o'zidan oldingi nusxadagiga qaraganda bitta kam bo'ladi. Ma'lum vaqtdan keyin n parametrning qiymati nol ga teng bo'ladi va f funksiyaga keying murojaat qilish bajarilmaydi. Funksiya birinchi nusxaga qaytgandan keyin dastur o'z ishini tamomlaydi.

Boshqa misol ko'ramiz:

```
Function ff(n:integer):integer;
Begin
If  n>0
```

Then ff:=ff(n+1)*n

Else ff:= 1

End.

Ff(n) ga murojaat qilishda uchragan dastur n>0 da chekli bo'lmaydi.

Ba'zi hollarda rekursiv funksiy va protseduralarning to'g'riligiga ishonch hosil qilishga ularda yozilgan amallarning masala ta'rifiga mos kelishi zarurligi yordam beradi.

Shunday qilib, dasturni tekshirish mashaqqatli mehnat ekanligi ravshan bo'ldi. Tajriba dasturning hammasi dastlabki ma'lumotlari mumkin bo'lgan har qanday qiymatlarida to'g'riligini isbot qilishga qaraganda ba'zi nazorat qiymatlar bo'yicha dastur natijalarining to'g'riligini tekshirish ancha soddaroq ekanligini ko'rsatadi. Ammo dastlabki ma'lumotlarning cheklangan miqdordagi variantlarda ba'zi xato natijalarni oson o'tkazib yuborish mumkin. bundan tashqari nazorat qiymatlarini tanlashda ham ma'lum qiyinchiliklar ham tug'iladi.

Dasturni tekshirishning quyidagi tartibini tavsiya qilish mumkin:

1. dastlabki ma'lumotlarning bitta yoki ikkita tanlab olingan variantlarida dasturning amallarini qo'lda bajarib, natijaviy qiymatlarni olish.
2. Tekshiriladigan dasturda sikllar, rekursiv funksiya hamda protseduralarning chekligini ko'rsatish.
3. Dastlabki ma'lumotlarning mumkin bo'lgan har qanday qiymatlarida dasturnin to'g'riligini isbotlashga harakat qilish. Agar buni bajarish ancha murakkab bo'lsa, unda bu dasturga xos bir nechta dastlabki ma'lumotlarni tanlab olish va shu qiymatlar bilan dasturdagi amallarni bajarish (ya'ni birinchi bosqichni nazorat qiymatlarning juda ko'p variantlarida takrorlash) kerak.

Dasturni takomillashtirish

Har doim ham birdaniga ixcham va tejamli dastur yaratishga muvofiq bo'linavermaydi. Ba'zida dastur tuzilgan, tekshirilgan bo'lsa-da, dasturchida dasturni takomillashtirishning yangi fikrlari (qisqaroq, aniqroq, lo'ndaroq yoki tejamliroq qilish) tug'ilib qoladi. Sonning mutlaq qiymatini aniqlovchi shartli operatorni misol tariqasida ko'rib chiqamiz:

```
If a>=0
    Then b:=0
    Else
     if a<0
    then b:=-a;
```

bu shartli operatorning ikkinchi qismida ortiqcha shartli operator yozilganini sezish qiyin emas, chunki a<0 shartni ikki marta tekshirish sodir bo'ladi.

Shartli operatorni qisqaroq va yaqqolroq yozamiz:

```
If a>=0
    Then b:=0
    Else b:=-a;
```

Boshqa misol ko'ramiz:

```
For k:=1 to 100 do
    Write(a+b);
```

Siklning har bir bajarilishida a va b o'zgaruvchilar qiymatlarini yig'indisi hisoblanadi va bosib chiqariladi. Yig'indisini hisoblashni sikldan tashqariga chiqaramiz:

```
C:=a+b;
For k:=1 to 100 do
Write(c);
```

Endi o’zgaruvchining qiymatlarini qo’shish faqat bir marta bajariladi. Demak, bu dasturni bajarishda mashina vaqti qisqaradi.

Kabisa yillarini aniqlovchi funksiyadagi shartli operatorni soddalashtirishga harakat qilamiz:

```
If yil mod 100 =0
    Then
If y mod 400=0
Then kabisa:=true
Else kabisa:=false
Else
If y mod 4 = 0
Then kabisa:=true
Else kabisa:=false;
```

Agar berilgan son 400 ga bo’linsa, bunday yil kabisa yili bo’ladi, shuning uchun oldin berilgan sonni 400 ga bo’linishini tekshirish, keyin esa qolgan hollarni ko’rib chiqish kerak:

```
    If y mod 400 = 0
    Then kabisa:=true
Else ……
```

Agar son 400 ga bo’linmasdan 100 ga bo’linsa bu son asrning oddiy yilini belgilaydi. Shartli operatorni yozishni davom ettiramiz:

If y mod 400 = 0

Then kabisa:=true

Else

if y mod 100=0

then kabisa:=false

else

……..

Qolgan yillar sonining 4 ga bo’linishini tekshiramiz va shartli operatorning oxirini yozamiz

If y mod 400=0

Then kabisa:=true

Else

If y mod 100 =0

Then kabisa:=false

Else

If y mod 4=0

Then kabisa:=true

Else kabisa:=false;

Shunday qilib, mulohazamiz natijasida shartli operator olinadi.

Dasturni tuzatish

Agar dastur yozilgan, sinchiklab tekshirilgan bo'lsa, uni mashinaga kiritish vaqti keldi. Endi dasturni EHM yordamida tuzatish zarur, chunki puxta tekshirilgan dasturda ham xatoliklar uchrashi mumkin. xatolar mazmuniy (semantik) , ko'p uchraydigan imloviy (Grammatik), shuningdek, sintaktik zarur belgilar yoki so'zlarni noto'g'ri ishlatish) xatolarga ajratiladi. Sintaktik xatolar dastur matnini kiritishda ham kelib chiqadi. Sintaktik xatolarni topib, EHM uning dasturdagi joyini dasturchiga xabar qiladi. Mazmuniy xatoni aniqlash ancha murakkab. Buning uchun, dasturning to'g'riligini qo'lda tekshirishdagiga o'xshash, tanlab olingan nazorat qiymatlarda tajriba o'tkaziladi. Xatolarni EHMning o'zi qidirayotganligi uchun, endi ancha «murakkab» ma'lumotlar bilan ish olib borish, sikllarning ko'p marta takrorlanishidan foydalanish mumkin.

Hamma mazmuniy xatolar ham tez va oson topilavermaydi. Murakkab dasturda chuqur yashiringan xatolar uchrashi mumkin. ularni topish uchun dasturchi sezgiga, mantiqan fikrlashga, yaxshi bilimga ega bo'lishi kerak.

Shunday qilib, dasturni tayyorlash va tuzatish dasturchidan anchagina mehnat talab qilar ekan. Biroq, dastur yaratilgandan keyin u bilan ko'p foydalanuvchilar ishlashi mumkin. zarur bo'lganda dastur matnini ko'chirish va turli xildagi EHM lar xotirasiga kiritish mumkin. **[5]**

2.2 Olimpiada masalalarini yechishga doir uslubiy tavsiyalar

Dasturlashga oid olimpiada masalalarini yechish metodikasining nazariyasiga binoan dasturlash masalalari turlicha klassifikasiyalanadi. Bunday masalalar quyidagi turlarga ajratiladi: Iqtisodiy masala, mantiqiy masala, matematik masala, biror bir muammoning yechimini topish masalalari, tadqiqot masalasi, va konstruktorlik, ya'ni ijodiy masala.

Dasturlashdan barcha o'quv masalalari masalaning mazmuniga, talab

xarakteriga, shartining berilishiga va yechish usuliga hamda maqsadiga qarab ham klassifikasiyalanadi. Bunda quyidagi metodik terminlar kiritiladi: matnli, hisoblash, eksperimental, mantiqiy, grafik, ijodiy va tadqiqot masalalari, konstruktorlik masalasi, muammoli masala, bilish va paradoksal, qiyin masalalar, demonstrasion masalalar, baholash masalalari, mazmuni rivojlanib boradigan. masalalar, frontal masalalar, javoblari tanlanadigan va mazmuni qisman o'xshash masalalar . Masalalarni yechish usuliga qarab, sifat va hisoblash masalalariga ajratiladi. Dasturlashga oid olimpiada masalasini yechish usuli deganda masala yechish jarayonini amalga oshirish uchun qo'llaniladigan konkret model tushuniladi.

Shunday qilib, informatikadan masalalar yechishda mantiqiy, matematik va eksperimental usullardan foydalaniladi. Matematik usul matematikaning qaysi bo'limidan foydalanilayotganiga qarab, quyidagi bir necha turga bo'linadi: arifmetik, algebraik, geometrik va grafik usullar, mantiqiy usul masalada berilgan holatni tushunishga va masalani sifat jihatdan yechishga imkon beradi. Masalalarni bunday klassifikasiyalash masalaning yaxlit ta'rifini kiritishga va o'quvchilarda masala yechishning umumlashgan o'quvini shakllantirish jarayonini maqsadga muvofiqlashtirishga imkon beradi.

Masalani yechish deyilganda, biror konkret masalani yechishni analiz qilish va yechishning asosiy bosqichlarini ajratish jarayonida ayrim usullarni ishlab chiqish tushuniladi. Ayrim turdagi masalalarni yechish metodikasini ishlab chiqish bilan masala yechish metodikasi takomillashib boradi. Bu, birinchi navbatda, hisoblash, grafik va eksperimental masalalarga tegishlidir. Bunda turli algoritmlardan foydalanishga e'tibor berish kerak. O'quv masalalarini yechishga algoritmik yondoshish o'quvchilarni fikrlash orqali matematik amallarni ajratishga, ma'lum turdagi masalalarni yechishda bajariladigan amallar sistemasini ishlab chiqarishga hamda yangi algoritmlar tuzish to'g'risidagi savolni yechishga imkon beradi. Odatda, dasturlash kursining turli temalari bo'yicha konkret masalalarni yechish namunalari keltiriladi. Shunday masalalarga sxematik o'xshash masalalarni yechishdagi o'quvchilarning mustaqil faoliyatlari ham masala yechish

deb qaraladi. Shuning uchun o’qituvchi uyga dastlabki topshiriq berishda shunga alohida e’tibor berishi kerak. Bu esa masala yechish qoidalarini va algoritmik amallarni bajarish sxemalarini yaratish zarurligini talab qiladi. Quyida mamlakatimizdagi ilg’or metodistlarning tajribalarini umumlashtirish asosida talaba taklif etgan masala yechish jarayoni qarab chiqiladi. Uning strukturasi,
ya’ni ayrim yoki umumiy amallarning mazmuni, masalaning shartiga binoan bo’ladi va tegishlicha talqin qilinadi. Masala yechish jarayonida o ’quvchilar har bir bosqichda quyidagilarni bajarishlari kerak.

-Birinchi bosqichda o’quvchi masala shartini o’qishda masalaning mazmunini yuzaki tushunib oladi va o’zida masalaning mohiyati haqida fikran tasavvur hosil qiladi.

-Ikkinchi va uchinchi bosqichda masalaning mazmuni ni to’laroq tasavvur qilish uchun masalani chizma yoki sxemaga qarab, qaytadan sekinroq o’qib chiqadi.

-To’rtinchi bosqichda masalani muvaffaqiyatli yechish uchun masala shartida berilgan ma’lum kattaliklarni aniqlaydi. Bu kattaliklarni yozish davomida O’quvchi masala shartini uchinchi marta o ’qiydi, bunda so’z va iboralar yoki gapning mazmuni bilan berilgan kattaliklarni aniqlaydi. Bunday kattaliklar
doimiylar, jadval ma ’lumotlari va biror hodisaning shartlari bo’lishi mumkin. Masalaning shartini qisqacha, qator yoki ustun ko’rinishda yozish mumkin. Masala sharti qisqacha yozilganda barcha berilgan o’zgaruvchilar, o’zgarmaslar, funksiyalar va protseduralar ularning harfiy belgilari orqali, son qiymatlari esa tegishli birliklari bilan birga yoziladi.

-Beshinchi va oltinchi bosqichda o’quvchi masalada qaralayotgan
hodisani tavsiflovchi algoritmik qonuniyatni masalaning mazmunidan
aniqlaydi.

O’quvchi masala yechishga kirishgan dastlabki paytda masala mazmunining mohiyatini, uning algoritmi sodda va tushunarli bo’lishini o’ylashi kerak. Shuning uchun o’quvchi Masalaning modelini ni hosil qilgandan so’ng,
sakkizinchi bosqichda masala shartida berilgan kattaliklarning dasturda xotira hajmini tejash maqsadida o’ziga qulay bo’lgan turni tanlashi kerak.

To’qqizinchi bosqichda masala shartida berilgan fizik kattaliklarni „ishchi formula" ga qo’ygan vaqtda ularning son qiymatlarini va birliklarini alohida-alohida yozib, barcha o’zgaruvchilarni o’z turiga mos qilib tanlash tavsiya etiladi. Bunday yozish esa masala yechimining to’g’riligini testlar orqali tekshirishga imkon beradi. Agar „ishchi formula" ga undagi har bir kattalikning. o’lchamligi to’g’ri tanlansa, izlanayotgan kattalikning o’lchamligi hosil bo’lishi kerak. Shundan so’nggina javobning son qiymatini hisoblashga kirishish kerak. O’rta maktablar uchun tavsiya etilgan informatikadan masalalar to’plamlaridagi o’quv masalalarida ko’p ma’lumotlar unchalik katta bo’lmagan aniqlikda beriladi. Shuning uchun barcha hisoblashlar ham taqriban o’tkazilib, masala shartida berilgan qiymatlarning aniqligi bilan chegaralaniladi. Oxirgi 11-bosqichda olingan javobning qiymati masalaning shartiga mos kelishi tekshiriladi. Sonlarni ortig’i bilan yoki kami bilan yaxlitlab hisoblashlar bajarish ba’zi hollarda masalaning javobiga keskin ta’sir etishi mumkin. Shuning uchun o’quvchi olingan javobning ishonchli va real ekanligiga ishonch hosil qilishi kerak. Turli bosqichdagi olimpiada masalalarining yechimini yozganda, bu ish tekshiruvchilarga qulay va tushunarli bo’lishi uchun o’quvchi masala yechishning borishidagi o’z fikr va mulohazalarini qisqacha bayon qilishi tavsiya etiladi. Bunday yozish. masala yechimining rejasiga muvofiq yaxlit ko’rinishda bo’lishi yoki har bir amalga qisqacha tushuntirishlar berish orqali bo’lishi mumkin.

dasturlashga oid olimpiada masalalarini yechishning eng umumiy usul va metodlari bayon qilingan, o’quvchilar uchun dasturlashdan chiziqli, tarmoqlanuvchi, takrorlanuvchi va shu kabi masalalar yechish tartibi ko’rsatilgan. Masalalarning shartlari batafsil tahlil qilingan va masalalarning yechimlari berilgan. Namunaviy masalalar va ularni yechish tartibi ko’rsatilgan.

2.3 Masalani turli xil metodlardan foydalanib yechish.

Massivlarni saralash usullari

Saralash masalasini yechishda, odatda, qo’shimcha xotiradan minimal foydalanish talabi suriladi, bundan qo’shimcha massivlardan foydalanish mumkin emasligi kelib chiqadi.

Saralashning har xil usullari algoritmlari ishi tezligini baholash uchun qoida bo’yicha, ikki ko’rsatkichdan foydalaniladi:

- O’zlashtirishlar soni;
- Taqqoslashlar soni;

Saralashning hamma usullarini ikki katta guruhga ajratish mumkin:

- Saralashning to’g’ri usullari;
- Saralashning yaxshilangan usullari;

Saralashning to’g’ri usullari o’z navbatida uch guruhga bo’linadi:

- Kiritish yo’li bilan saralash;
- Tanlash yo’li bilan saralash;
- Almashtirish yo’li bilan saralash («ko’pik» usuli);

Saralashning yaxshilangan usullari ham to’g’ri usullar tayangan tamoyilga asoslanadi, lekin saralashni tezlashtiradigan ba’zi bir g’oyalardan foydalanadi. Amalda to’g’ri usullar, past tezlikka ega bo’lganligi uchun, kam ishlatiladi. Lekin to’g’ri usullar ularga asoslangan yaxshilangan usullarning mohiyatini yaxshi ko’rsatadi. Bundan tashqari, ba’zi bir hollarda uzun bo’lmagan massivda va (yoki) massiv elementlarining o’ziga xos joylashgan vaqtida to’g’ri usullarning ba’zilari yaxshilangan usullardan ham o’tishi mumkin.

Kiritish yo’li bilan saralash

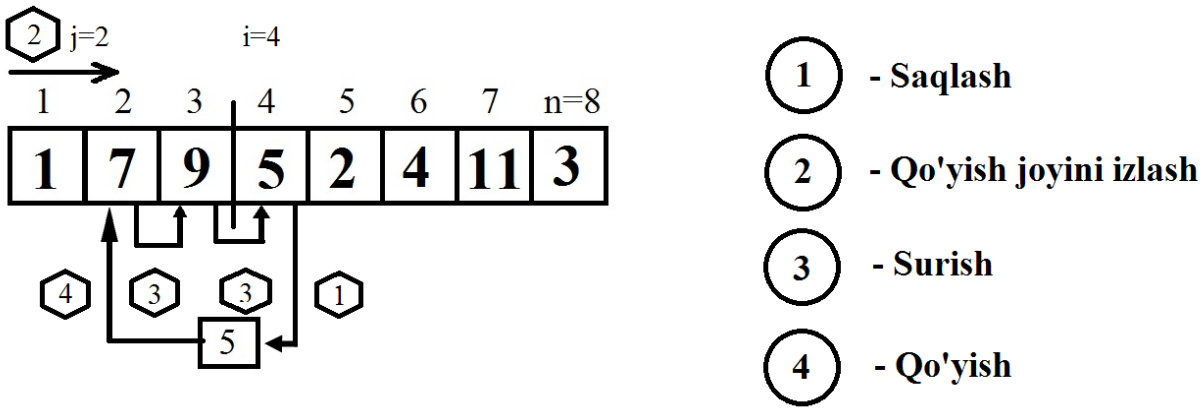
Usul tamoyili. Massiv saralangan va saralanmagan ikki qismga bo’linadi: Saralanmagan qismdan elementlar navbat bilan tanlanadi va saralangan qismga, elementlar tartibini buzmasdan, kiritiladi. Algoritm ishi boshida massivning saralangan qismi sifatida faqat bitta birinchi element, saralanmagan qism sifatida esa qolgan elementlar olinadi.

Shunday qilib, algoritm har biri to’rtta ishdan iborat n-1 kiritishdan (massiv n o’lchamli) tashkil topadi:

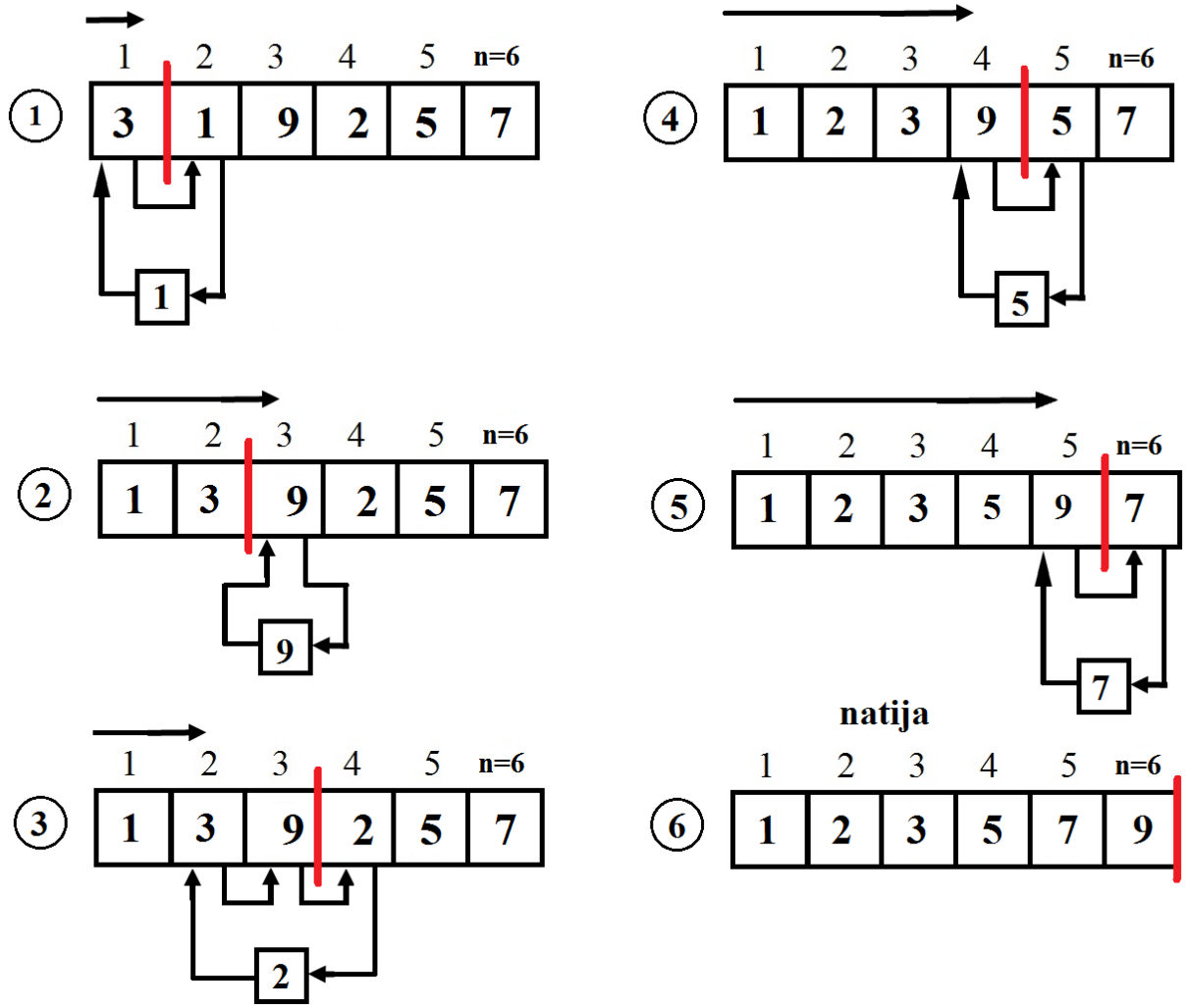
- Navbatdagi i – saralanmagan elementni olish va uni qo’shimcha o’zgaruvchida saqlash;
- Massivning saralangan qismida olingan elementning mavjudligi elementlar tartibini buzmaydigan, j – o’rinni izlash;
- Topilgan o’rinni bo’shatish uchun massiv elementlarini i – 1 dan j – 1 gacha o’ngga siljitish;
- Olingan elementni topilgan j – o’ringa joylashtirish.

Bu usulni amalga oshirish uchun bir – biridan kiritish o’rnini izlash usullari bilan farq qiluvchi, bir nechta algoritmni taklif qilish mumkin. mumkin bo’lgan algoritmlardan birini amalga oshirish chizmasini ko’ramiz.

Bitta kiritish ishini chizma sifatida quyidagicha bayon etish mumkin:



Kiritish yo’li bilan saralash chizmasi. Chapda aylana ichida o’tish raqami ko’rsatilgan:



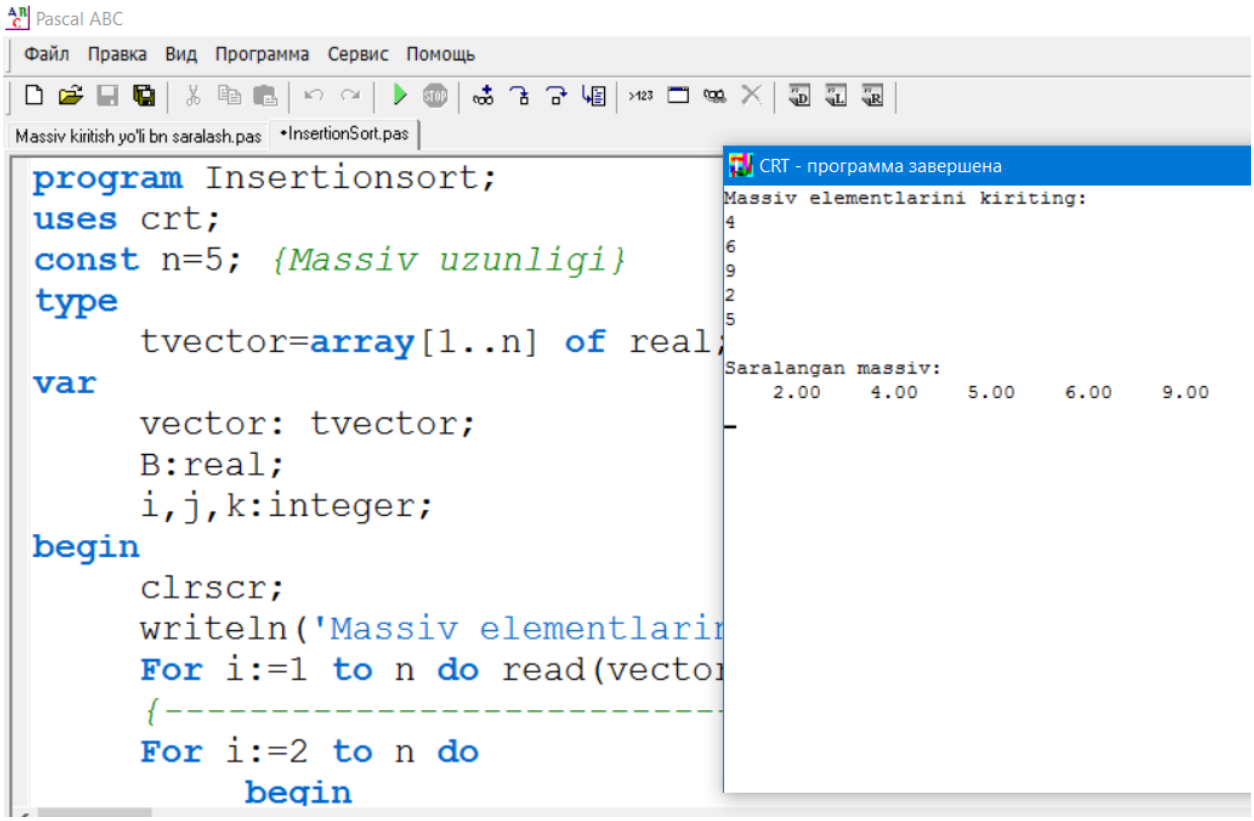
Ko’rib chiqilgan algoritmni amalga oshirish dasturini keltiramiz:

```
program InsertionSort;

uses crt;

const n=5; {Massiv uzunligi}

type

    tvector=array[1..n] of real;

var
```

```
    vector: tvector;
    B:real;
    i,j,k:integer;
begin
    clrscr;
    writeln('Massiv elementlarini kiriting: ');
    For i:=1 to n do read(vector[i]); readln;
    {--------------------------------------}
    For i:=2 to n do
        begin
        B:=vector[i]; {saralanmagan elementni olish}
                {qo'yish o'rnini topish sikli}
        j:=1;
        while (B>Vector[j]) do
        j:=j+1; {j indeks sikl tugagandan keyin }
              {qo'yish o'rnini belgilaydi.}

   {Qo'yish o'rnini bo'shatish uchun elementlarni}
   {siljitish sikli }
   for k:=i-1 downto j do
   vector[k+1]:=vector[k];
```

```
    {topilgan o'ringa olingan elementni kiritish}
    vector[j]:=B;
    end;
    {-----------------------------------------}
    writeln('Saralangan massiv:');
    for i:=1 to n do write(vector[i]:8:2);
    writeln;
end.                    [5].
```

Dastur kodini Pascal ABC dasturiga kiritib, kompilyatsiya qilamiz.



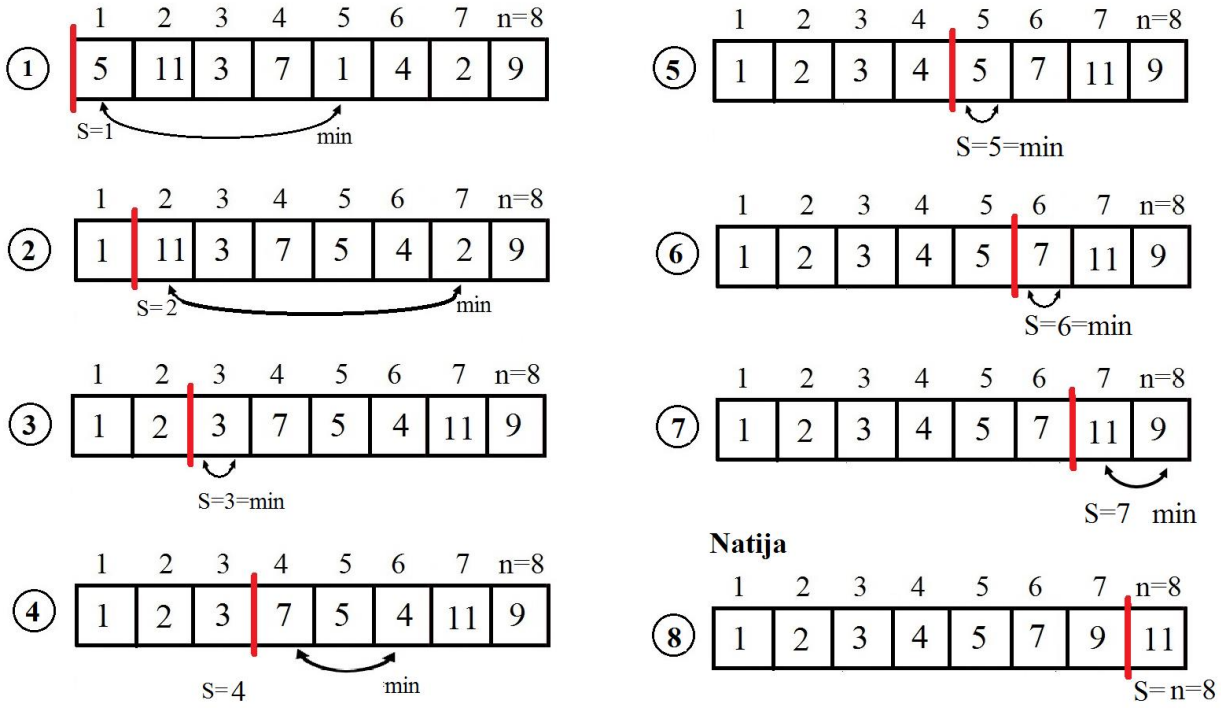
Tanlash yo’li bilan saralash

Usul tamoyili. Mas sivda 1 – elementdan n – elementgacha (oxirgi) bo’lgan oraliqda minimal qiymatli elementni topamiz (tanlaymiz) va uni birinchi element

o’rni bilan almashtiramiz. Ikkinchi qadamda 2 – dan n – gacha oraliqda minimal qiymatli elementni topamiz va uni ikkinchi element o’rni bilan almashtiramiz.

Shunday tarzda n-1 elementgacha almashtirishlarni bajaramiz.

Tanlash algoritmi chizmasini ko’ramiz:

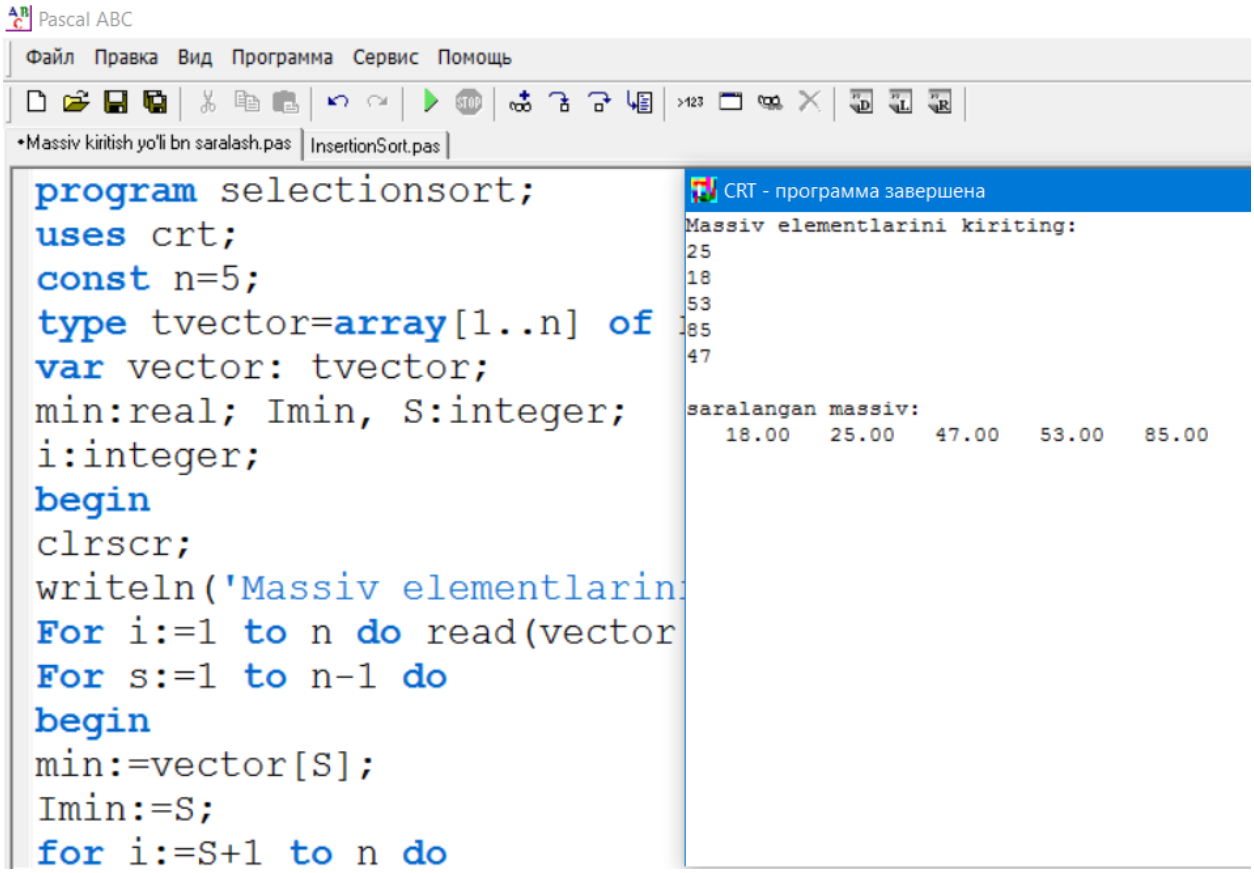


Tanlash usulini amalga oshiruvchi dastur matnini keltiramiz:

```
program SelectionSort;

uses crt;

const n=5;

type tvector=array[1..n] of real;

var vector: tvector;

min:real; Imin, S:integer;

i:integer;

Begin
```

```
clrscr;
writeln('Massiv elementlarini kiriting: ');
for i:=1 to n do read(vector[i]); readln;
For s:=1 to n-1 do
        begin
           min:=vector[S];
           Imin:=S;
           for i:=S+1 to n do
           if vector[i]<Min then
                begin
                        min:=vector[i];
                        Imin:=i;
                end;
           Vector[Imin]:=vector[s];
           vector[s]:=min;
        end;
writeln('saralangan massiv:  ');
for i:=1 to n do
        write(vector[i]:8:2);
        writeln;
end.                    [5].
```

Dastur kodini Pascal ABC dasturiga kiritib, kompilyatsiya qilamiz.



Almashtirish yo’li («BubleSort» usuli) bilan saralash

Usul tamoyili. Chapdan o’ngga tomon navbat bilan ikki qo’shni element, tartibga solishning berilgan shartiga mos kelmasa, o’rinlari bilan almashtiriladi. Shundan keyin navbatdagi qo’shni elementlar olinadi, xuddi shunday tarzda, massiv oxirigacha, ish bajariladi.

Bir marta shunday o’tishdan keyin massivning oxirgi n – xonasida maksimal element joylashgan bo’ladi (birinchi «ko’pik» yuzaga chiqadi). Oxirgi element o’zining oxirgi xonasida turgani uchun, almashtirishning ikkinchi o’tishi endi n-1 elementgacha bo’ladi va h.k. Hammasi bo’lib, n-1 ta o’tish talab qilinadi.

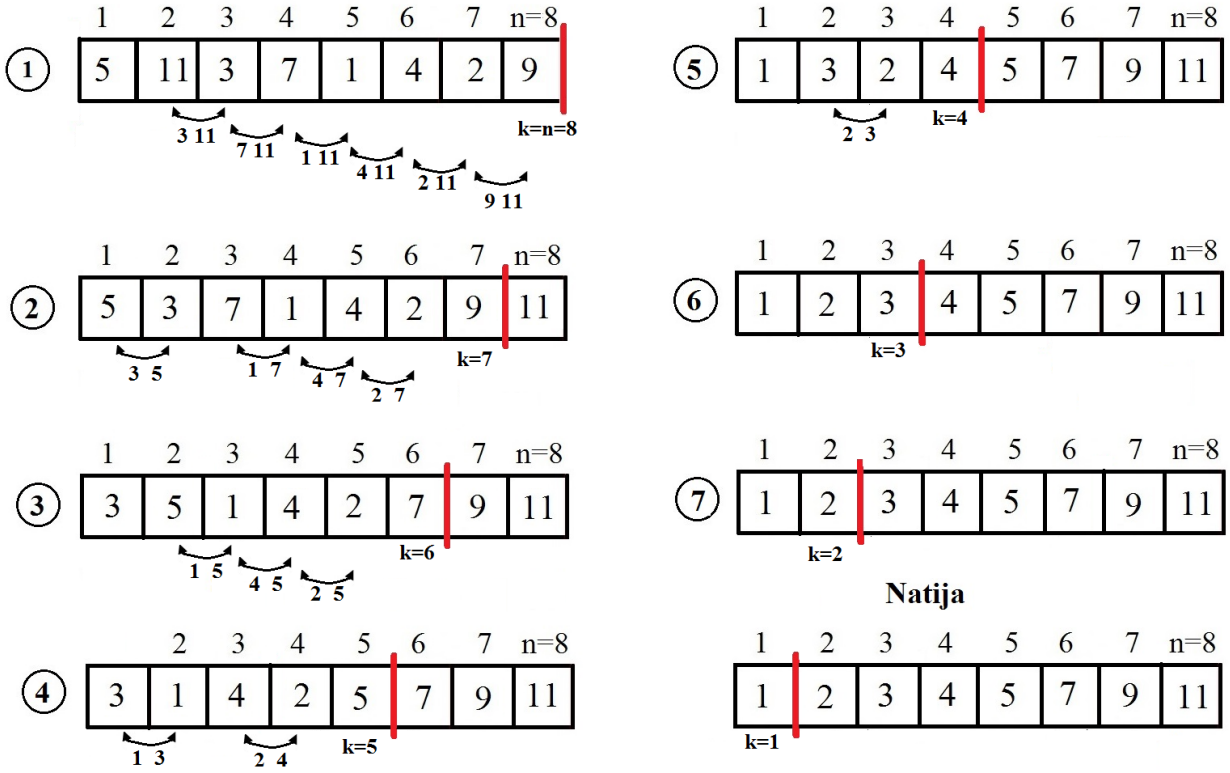
Pufaksimon usulni massiv elementlarida pastdan yuqoriga va yuqoridan pastga o’tishini bir vaqtda amalga oshirish natijasida yaxshilash mumkin.

Taqqoslashlar soni M=(n/2)* (n/2)=$n^2/4$

Almashtirishlar soni :

C=3* $n^2/4$

Almashtirish usulining o’sib borish tartibida saralash algoritmi chizmasini ko’ramiz:

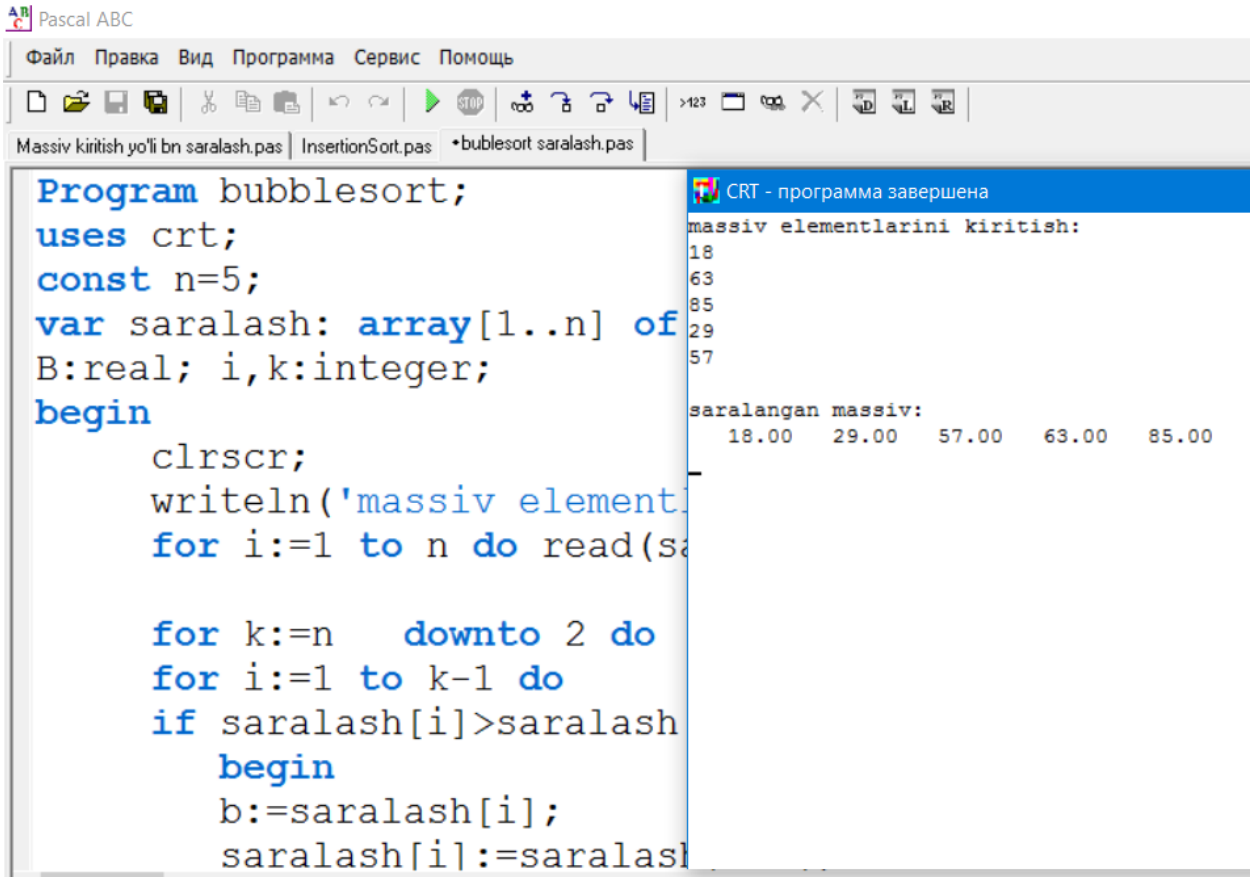


Dastur matnini keltiramiz:

```
Program BubbleSort;

uses crt;

const n=5;

var saralash: array[1..n] of real;

B:real; i,k:integer;
```

```
Begin
    clrscr;
    writeln('massiv elementlarini kiriting:');
    for i:=1 to n do read(saralash[i]); readln;
{-------------------------------------------------------------}
    for k:=n   downto 2 do
    for i:=1 to k-1 do
    if saralash[i]>saralash[i+1] then
      begin
        b:=saralash[i];
                saralash[i]:=saralash[i+1];
       saralash[i+1]:=b;
      end;
{-----------------------------------------------------------------------------}
    writeln('saralangan massiv:');
    for i:=1 to n do write(saralash[i]:8:2);
    writeln
end.                          [5].
```

Dastur kodini Pascal ABC dasturiga kiritib, kompilyatsiya qilamiz.



Saralash usullarini taqqoslash

Saralash usullari algoritmlarini nazariy va amaliy tadqiq qilish shuni ko’rsatadiki, taqqoslash, shuningdek, o’zlashtirish soniga ko’ra ular n massiv uzunligiga bog’liq. O’zlashtirish tartibi n*ln(n) soniga teng bo’lgan tanlash usuli bundan mustasno. Tanlash algoritmining bu xossasini bitta taqqoslashda katta sondagi o’zlashtirishlar bajariladigan murakkab tasnifli ma’lumotlarni saralash masalalarida qo’llash foydali. Bunday masalalarda tanlash usuli saralashning eng tez yaxshilangan usullari bilan bahslasha oladi.

Ko’rib o’tilgan to’g’ri usullar o’zaro taqqoslansa, kiritish va tanlash usullari o’rtacha taqriban teng kuchli va almashtirish (“ko’piklar”) usuliga qaraganda bir necha marta (massiv uzunligiga bog’liq ravishda) yaxshi.

Palindrom sonlarni topish masalasi

O’ngdan chapga va chapdan o’nga bir xil o’qiladigan sonlar ***palindromlar*** deyiladi. Masalan 42324, 585 yoki 1661 sonlari – palindrom. Berilgan oraliq ichida palindrom sonlarni topuvchi dastur tuzing.

1-usul:

Dasturni tuzish mantig’i quyidagicha:

Sondagi raqamlar o’rnini almashtirish va hosil bo’lgan sonni berilgan son bilan solishtirish.

repeat ... **until** ...

sikli yordamida qanday qurilishini ko’raylik.

a son berilgan bo’lsa, u holda yana bir **b** o’zgaruvchini kiritamiz, unga **a** o’zgaruvchining qiymati beriladi b:= a;

Yana bir o’zgaruvchi **a1** ni raqamlarning o’rnilari almashtirilib bo’lingan yangi son uchun hosil qilamiz. Ushbu o’zgaruvchining boshlang’ich qiymati - nol: a1 := 0;

Bu o’zgaruvchi nima uchun nolga tengligi dasturni ko’rib chiqqanimizdan so’ng ravshan bo’ladi.

So’ngra, **b** sondagi raqamlarning o’rin almashish jarayoni kechadigan **repeat** siklini tashkil qilamiz:

```
repeat
  a1 := a1*10 + b mod 10;
  b  := b div 10
until b = 0;
```

Shunday qilib, siklda **while** ... **do** ... siklidagidek, oxirgi raqam ajratiladi:

b **mod** 10; (masalan, 343 **mod** 10 = 3); **a1** o'zgaruvchiga quyidagi qiymat beriladi:

a1 := a1*10 + b **mod** 10; 0 * 10 + 3 =3;

Berilgan sondagi oxirgi raqam butun bo'lish operatsiyasi yordamida "*tashlab yuboriladi* " :

b := b **div** 10; 343 **div** 10 = 34;

b = 0 ya'ni 34 = 0 shart tekshiriladi, shart bajarilmayapti, demak sikl davom etadi.

Yangi sondagi oxirgi raqam ajratiladi:

b **mod** 10 = 34 **mod** 10;

3 ga teng bo'lgan yangi **a1** son, 10 ga ko'paytiriladi va natijaga keyingi raqam - 4 qo'shiladi:

a1 := a1*10 + b **mod** 10;

b sondagi oxirgi raqam "*tashlab yuboriladi*":

b := b **div** 10 ; 34 **div** 10 = 3;

b = 0, 3 = 0 shart tekshiriladi; shart bajarilmayapti, demak sikl davom etadi.

Sonning oxirgi raqami ajratiladi:

b **mod** 10 ; 3 **mod** 10 = 3;

yangi son hosil bo'ladi:

a1 := a1*10 + b **mod** 10 ; 34 * 10 + 3 = 343;

sondagi oxirgi raqam "*tashlab yuboriladi*" va yangi son hosil bo'ladi:

b := b **div** 10 ; 3 **div** 10 = 0;

b = 0, 0 = 0 shart tekshiriladi; shart bajarilmoqda, demak sikl tugatiladi.

Endi, berilgan son uchun nega boshqa **b** o'zgaruvchi kiritilgani tushunarli, sonning sikldagi qiymati boshlang'ich qiymatdan boshlab 0 gacha o'zgaradi va **a** o'zgaruvchida berilgan sonni saqlash uchun *"ishchi"* o'zgaruvchi - **b** kiritiladi.

Sondagi raqamlar o'rnini almashtirish sikli tugatilgandan so'ng, **a** o'zgaruvchida "*saqlab qolingan*" boshlang'ich son va raqamlar o'rnini amashtirgandan so'ng hosil bo'lib, **a1** o'zgaruvchida "*turgan*" son solishtiriladi.

Agar a = a1 bo'lsa, ***u holda*** a ning qiymati ekranga chiqariladi, chunki bu son ***palindromdir.***

So'ngra, **a** ning qiymati **1** ga oshadi, ya'ni ko'rib chiqish uchun keyingi natural son olinadi va yana tashqi sikl davom etadi. Sondagi raqamlar o'rni almashtiriladi, raqamlar o'rnini almashtirgandan so'ng hosil bo'lgan yangi **a1** son boshlang'ich **a** bilan solishtiriladi va h.k.

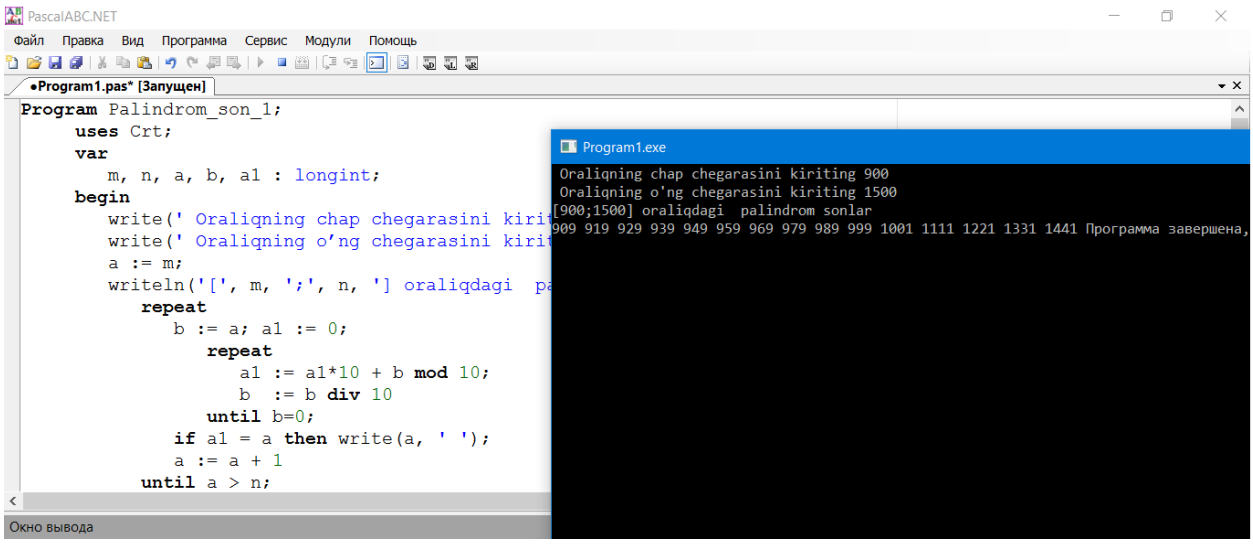
a ning qiymati **n** oraliqning o'ng chegarasiga teng bo'lib qolganida tashqi sikl tugatiladi.

Dastur tuzamiz

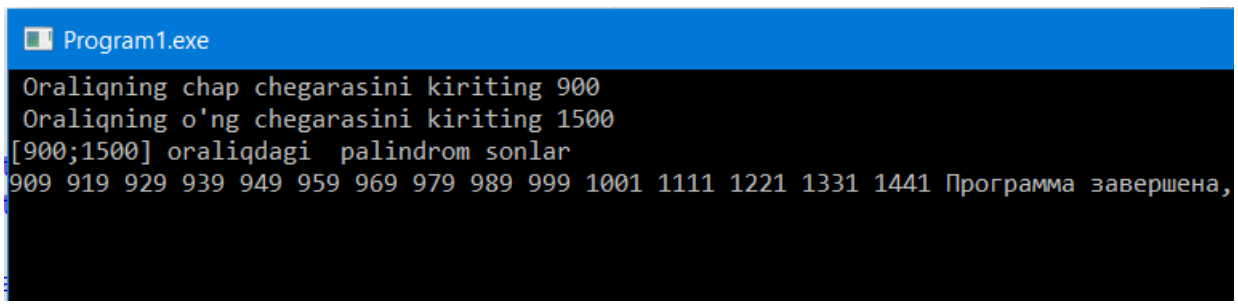
```
Program Palindrom_son_1;
     uses Crt;
     var
        m, n, a, b, a1 : longint;
     begin
        write(' Oraliqning chap chegarasini kiriting ');
readln(m);
        write(' Oraliqning o'ng chegarasini kiriting ');
readln(n);
        a := m;
```

```
        writeln('[', m, ';', n, '] oraliqdagi  palindrom
sonlar');
            repeat
               b := a; a1 := 0;
                  repeat
                     a1 := a1*10 + b mod 10;
                     b  := b div 10
                  until b=0;
               if a1 = a then write(a, ' ');
               a := a + 1
            until a > n;
      end.
```

Dasturni PascalABC . NET dasturiga kiritamiz va oraliqning chap va o'ng chegarasini kiritamiz, masalan [900 ; 1500]



Quyida dastur natijasining kattalashtirilgan holatdagi ko'rinishi.



2- usuli: Palindrom sonlarni topuvchi funksiya yozamiz.

funksiyaning qiymati mantiqiy ya'ni sonni palindrom ekanligini tekshiradi.

function isPalindrom(n:longint):boolean;

va quyidagi blok ichida funksiya amallari bajariladi.

Begin

str(n,s); {satrdan butunga o'tkazish}

len:=Length(s) **div** 2; - len o'zgaruvchiga s sonining raqamlar sonini 2 ga bo'lib butun qismini o'zlashtiradi.

inc:=Length(s) **mod** 2; - Inc o'zgaruvchiga s sonining raqamlar sonini 2 ga bo'lib qoldiq qismini o'zlashtiradi.

Result:=Copy(s, 1, len)=ReverseString((copy(s,len+inc+1,len)));

{Result funksiyadan natija qaytaradi} – yuqoridagi satrda s sonning 1-belgisidan boshlab **len** o'zgaruvchimizning qiymatiga teng bo'lgan joygacha raqamlardan nusxa oladi, ReverseString operatorimiz esa s sonning yarmidan boshlab oxirigi raqamigacha bo'lgan sondan nusxa oladi va sonning raqamlari joyini almashtiradi masalan: 123 sonini 321 qilib beradi Va mantiqiy tenglashtiriladi. Agar ular teng bo'lsa funksiya natija qaytaradi. Kompyuter bu sonlarning palindrom ekanligini tushunmaydi. Shunchaki sonning o'rtasidan chap tarafdagi raqamlari bilan o'ng tomondagi raqamlari joylashish tartibi bir xil bo'lsa masalan: 585, 1001, 123321 yoki 12321 bo'lsa shu sonlarni natija sifatida chop etadi. Funksiyamizni yozib bo'lganimizdan keyin dasturning tana qismiga o'tamiz va oraliqning chap va o'ng chegarasini kiritamiz. Oraliqni kiritib bo'lgandan so'ng For (takrorlash) operatori [m; n] oraliqdagi sonlarni Palindrom ekanligini tekshirib chiqadi agar palindrom bo'lsa mantiqiy true (rost) qiymat qaytaradi va ekranga chiqaradi, agar shart qanoatlanmasa u sonlarni chop etmaydi, va keying songa o'tadi va h.k.

Dasturning asosiy qismi

Begin

write('oraliqning chap chegarasini kiriting '); readln(m);

write('oraliqning ong chegarasini kiriting '); readln(n);

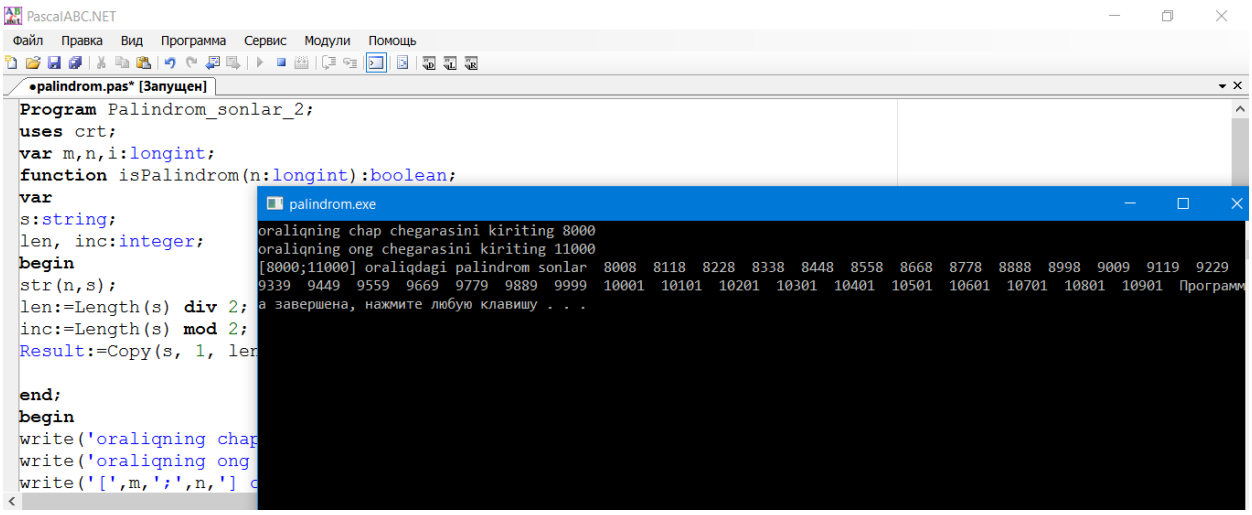
```
write('[',m,';',n,'] oraliqdagi palindrom sonlar  ');
for i:=m to n do
   if isPalindrom(i)=true then write(i,'  ');
End.
```

Algoritmning ishlash strukturasi shunday tartibda amalga oshadi.

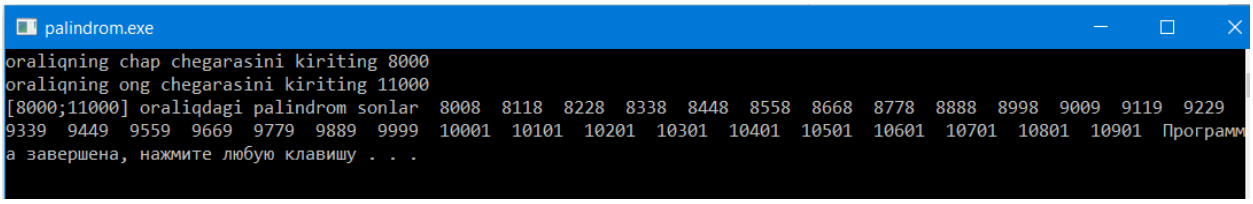
Dastur tuzamiz:

```
Program Palindrom_sonlar_2;
uses crt;
var m,n,i:longint;
function isPalindrom(n:longint):boolean;
var
s:string;
len, inc:integer;
begin
str(n,s);   {satrdan butunga o’tkazish}
len:=Length(s) div 2;
inc:=Length(s) mod 2;
Result:=Copy(s, 1, len)=ReverseString((copy(s,len+inc+1,len)));
{Result funksiyadan natija qaytaradi}
end;
begin
write('oraliqning chap chegarasini kiriting '); readln(m);
write('oraliqning ong chegarasini kiriting '); readln(n);
write('[',m,';',n,'] oraliqdagi palindrom sonlar  ');
for i:=m to n do
   if isPalindrom(i)=true then write(i,'  ');
end.
```

Dasturni PascalABC . NET dasturiga kiritamiz va oraliqning chap va o’ng chegarasini kiritamiz, masalan [8000 ; 11000]



Quyida dastur natijasining kattalashtirilgan holatdagi ko’rinishi



Palindrom sonlarni topish usullarini taqqoslash

1-usulda - Repeat operatoridan foydalanilgan. Bu dasturda ikkita sikl ichma-ich joylashgan bo’lib, bitta sonni palindrom ekanligini aniqlash uchun sikl shart rost bo’lguncha bajarilaveradi. sonni avval ichki sikl rost qiymat qabul qiluncha qayta-qayta hisoblab tekshiraveradi. Agar shart rost bo’lsa ya’ni b=0 bo’lsa tashqi sikl bajariladi. Undan keyin a1=a yangi hosil bo’lgan son bilan avvalgi son tengligi tekshiriladi. Agar teng bo’lsa natija sifatida chop etadi va navbatdagi songa o’tadi ya’ni a:=a+1 Bu dasturda bitta sonning palindrom ekanligi aniqlash uchun dastur bajaradigan amallar (qadamlari) soni ko’proq.

2-usulda - dastur qismiga fuksiya yozilgan bo’lib, funksiya orqali sonlarning palindrom ekanligi tekshiriladi. Yani berilgan sonni satrdan butunga o’tkazadi (sababi satrni reverse qilish operatoridan foydalanganimiz uchun) sonni ikkiga bo’lib butun qismini len o’zgaruvchiga o’zlashtiradi, s sonni yana ikkiga bo’lib

qoldiq qismini olib inc o’zgaruvchiga o’zlashtirib beriladi. S sonning birinchi raqamidan yarmigacha nusxa olib qolgan yarmining nusxasini teskari tartibda joylashtirib ularni mantiqan tengligini tekshiradi agar rost bo’lsa for operatori siklida natijani chop etadi. Bu usulda qadamlar soni 1-usulga nisbatan kamroq bajariladi va biz vaqtdan va bajariladigan amallar sonidan yutamiz. Palindrom sonlarni chiqarishda bu usuldan foydalanish afzal.

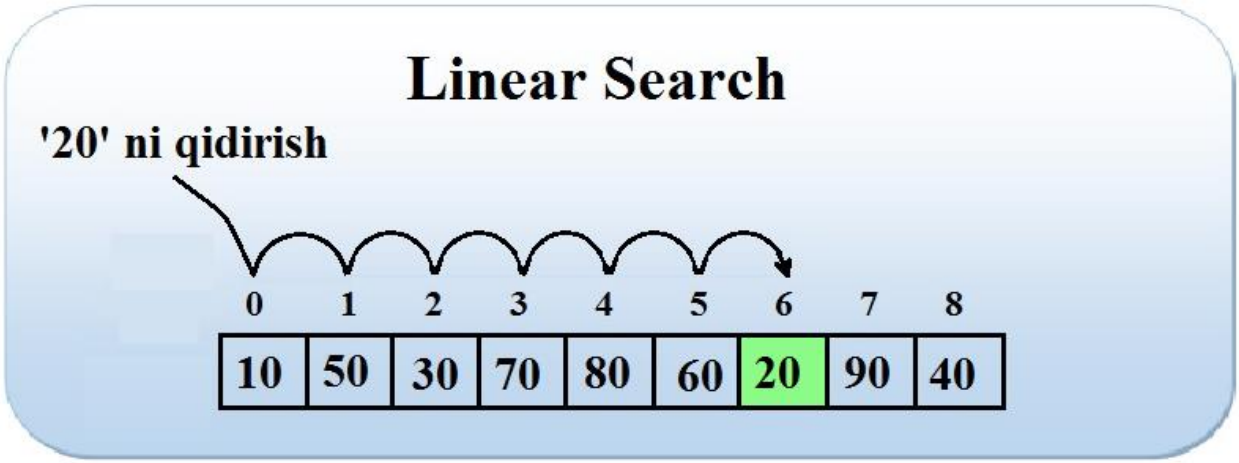
Massiv elementlarini qidirish

Massivlarda qidirish (massivda qidiruv, chiziqli qidiruv, binar qidiruv). Agar massiv saralangan bo`lsa, massivdan biror elementni topish uchun chiziqli qidiruvdan ko`ra ikkilik qidiruvi ko`proq samara beradi.

Massivda qidiruv – kerakli elementni topish jarayonidir. Masalan, aniq bir qiymat turining massiv tarkibidagi elementlardan birida mavjudligini tekshirish. Kompyuter dasturlashda eng ko`p qidiruv masalasi uchraydi. Qidirishni amalga oshirishga yo`naltirilgan ko`plab algoritmlar va ma’lumot tuzilmalari mavjud. Bu qismda qidirishni amalga oshirishning ikki xil yondoshuvlari – chiziqli qidiruv va ikkilik qidiruv o`rganiladi. Chiziqli qidirish yondoshuvi kalit – kalit so`zini massivning har bir elementi bilan, ketma-ket solishtirishga ixtisoslashtirilgan. Funksiya bu ishni massiv tarkibidagi elementlardan birontasi kalit ga mos kelgunicha yoki, massiv so`nggiga qadar amalga oshiradi. Agar massiv tarkibida biror element kalit ga mos kelsa, chiziqli qidirish uning indeksini qaytaradi. Aks holda, qidiruv -1 ni qaytaradi. Bir qancha qiymatlarni qidirishning eng ko`p tarqalgan yodoshuvlaridan biri – ikkilik qidiruv yondoshuvidir. U massivdagi elementlar avvaldan tartiblangan bo`lishini talab qiladi. Tasavvur qiling, massiv o`sish tartibida. Ikkilik qidiruv birinchi navbatda kalit so`zni massivning markazidagi element bilan solishtiradi. Quyidagi holatlarni ko`rib chiqamiz:

- Agar kalit markaziy elementdan kichik bo`lsa, qidirish massivning faqat birinchi yarimtaligida davom ettiriladi;
- Agar kalit markaziy elementga mos bo`lsa, qidirish brinchi urinishdayoq to`xtatiladi;
- Agar kalit markaziy elementdan katta bo`lsa, qidirish massivning faqat ikkinchi yarimtaligida davom ettiriladi.

1-usul: Chiziqli qidiruv (Linear search) – bunda berilgan kalit chiziqli ketma ketlikda to’plam elementlari orasidan qidirib chiqiladi. Bu algoritm ancha sodda lekin sekin hisoblanadi.

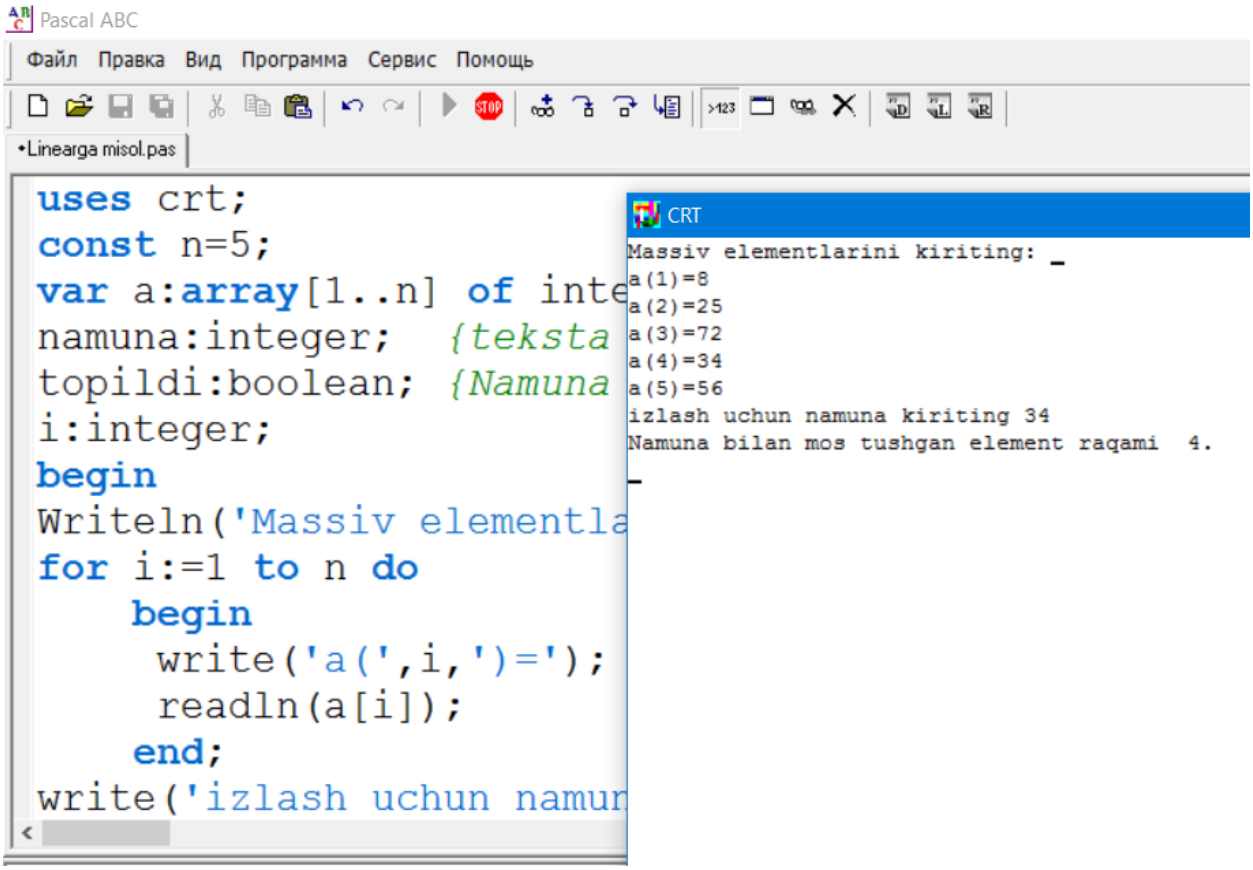


Dasturimizni Pascal dasturiga kiritamiz va Natijani tahlil qilib ko’ramiz.

```
uses crt;

const n=5;

var a:array[1..n] of integer;

namuna:integer;  {tekst izlash uchun namuna}

topildi:boolean; {Namuna bilan moslik belgisi}

i:integer;

begin
```

```
Writeln('Massiv elementlarini kiriting: ');
for i:=1 to n do
    begin
     write('a(',i,')=');
     readln(a[i]);
    end;
write('izlash uchun namuna kiriting ');
readln(namuna);
topildi:=false; {moslik yo`q}
i:=1;   {Massivning 1-elementidan boshlab tekshiramiz}
repeat
      if a[i]=namuna
       then topildi:=true  {namuna bilan mos tushdi}
      else i:=i+1;  {aks xolda navbatdagi elementga o'tish}
until (topildi) or (i>n);
if topildi
   then writeln('Namuna bilan mos tushgan element raqami',i:3,'.')
   else writeln('Namuna bilan moslik yo`q');
  readln;
end.       [6].
```

Dastur kodini Pascal ABC dasturiga kiritamiz va kompilyatsiya qilamiz;



2-usul: ***Ikkilamchi izlash (binary search) , ikkiga bo'lish bilan izlash***

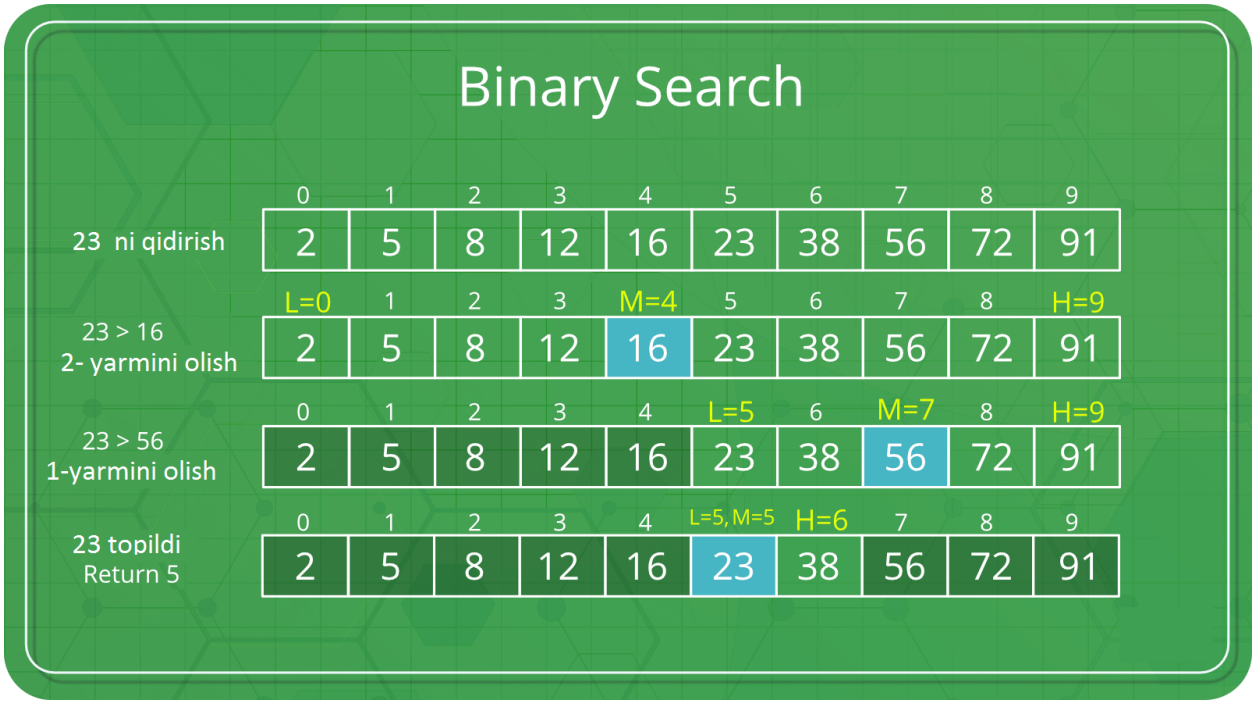
Ikkilamchi izlash usuli algoritmini berilgan elementni faqat tartibga solingan massivlarda qo'llab bo'ladi. Uni o'sib borish tarzida tartibga solingan massiv misolida ko'ramiz (vector[i]<=vector[i+1]).

Ikkilamchi izlash usuli tamoyili

Boshlang'ich massiv ikkiga bo'linadi va taqqoslash uchun o'rta element olinadi. Agar u izlanayotgan element bilan mos kelsa, izlash tugaydi. Agar o'rtacha element izlanayotganidan kichik bo'lsa, undan chapda yotgan elementlarning hammasi ham izlanayotganidan kichik bo'ladi. Demak, ularni, massivning faqat o'ng tomon qismini qoldirib, keyingi izlash sohasidan o'chirib tashlash mumkin. xuddi shunday, agar o'rta element izlanayotganidan katta bo'lsa, massivning chap tomoni saqlanib, o'ng tomoni olib tashlanadi.

Ikkinchi bosqichda massivning qolgan yarmi bilan xuddi shunday ishlar bajariladi. Ikkichi bosqich natijasida massivning ¼ qismi qoladi. Bu ish, element topilmaguncha, yoki izlash sohasi nolga teng bo'lmaguncha davom ettiriladi. Oxirgi holatda izlanayotgan element topilmaydi.

Ikkilik qidiruv algoritmi – Bu algoritm asosan to'plamni ikkiga bo'lishlar orqali qidirishdan iborat. Yani unda bo'linishlar toki kalit so'z topilmagunicha davom etadi.



L *– Massiv elementlarining chap indeks raqami.*

H *– Massiv elementlarining o'ng indeks raqami.*

M *– Massivning o'rta elementi indeks raqami*

Ikkilamchi izlash usulini amalga oshiruvchi dasturning mumkin bo'lgan variantlaridan birini keltiramiz:

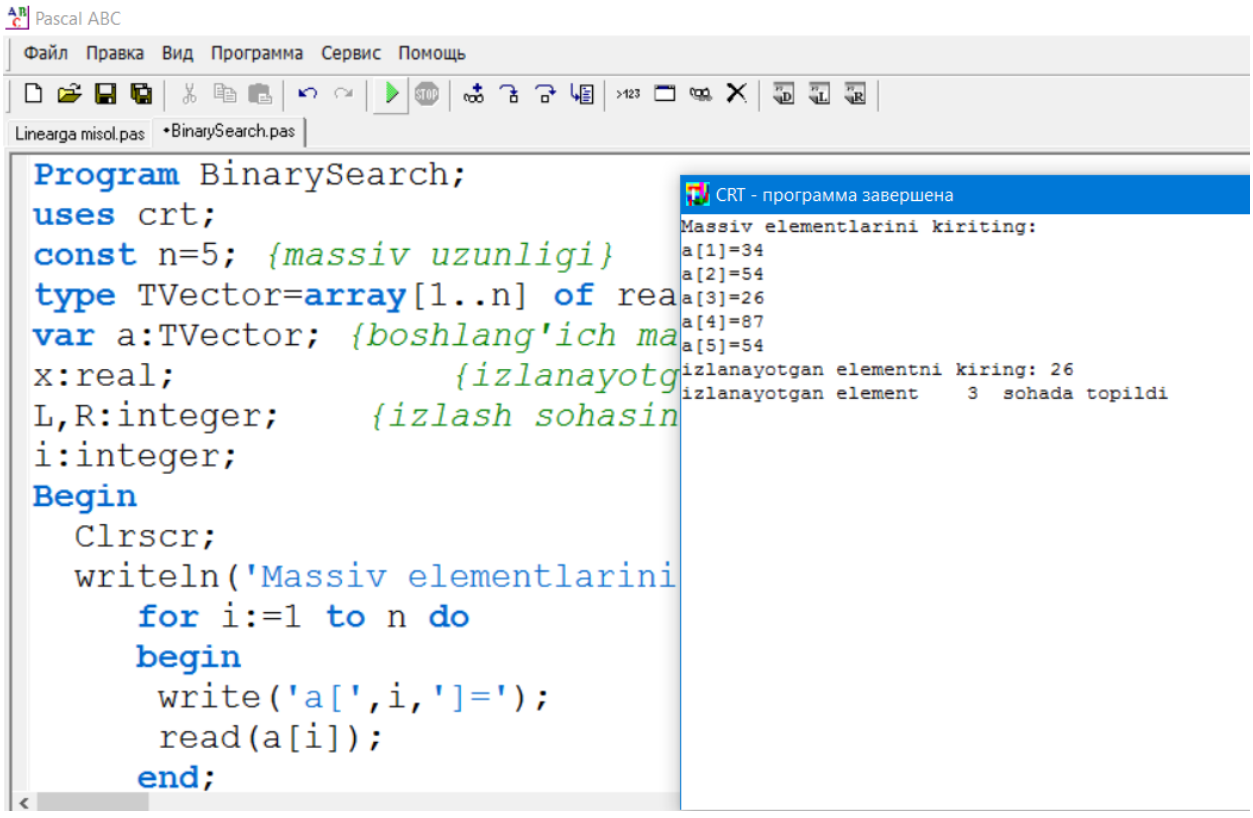
```
Program BinarySearch;

uses crt;
```

```
const n=5; {massiv uzunligi}
type TVector=array[1..n] of real;
var a:TVector; {boshlang'ich massiv}
x:real;            {izlanayotgan element}
L,R:integer;    {izlash sohasining joriy chegaralari}
i:integer;
Begin
  Clrscr;
  writeln('Massiv elementlarini kiriting: ');
     for i:=1 to n do
     begin
      write('a[',i,']=');
      read(a[i]);
     end;
Write('izlanayotgan elementni kiring: ');
Readln(x);
L:=1; R:=n;
while (L<=R) do   {chegaralar kesishmaguncha}
      begin
       i:=(L+R) div 2; {o'rta element indeksi}
       if a[i]=X then
```

```
      Break  {Izlash siklidan chiqish, chunki element topildi}
       else
       if a[i]<X then L:=i+1
              else R:=i-1;
    end; {while}
if a[i]=X then
writeln('izlanayotgan element  ',i:3, '  sohada topildi')
else
writeln('izlanayotgan element topilmadi');
end.
```

[5].

Dastur kodini Pascal ABC dasturiga kiritamiz va kompilyatsiya qilamiz



Qidirish usullarini taqqoslash

Bu ikkala usul Linear Search va Binary Search qidirish usullaridan foydalanuvchi berilgan massivning tartiblangan yoki tartiblanmaganligiga qarab foydalanishi maqsadga muvofiq. Agar massiv elementlari tartibsiz joylashgan bo’lsa u holda Linear Search (chiziqli qidirish) afzal. Agar massiv elementlari tartibli joylashgan bo’lsa Binary search (ikkiga bo’lib qidirish) usulidan foydalanish yaxshi natija beradi.

XULOSA

O’zbekiston Respublikasi Kadrlar tayyorlash milliy dasturi va «Ta’lim to’g’risida»gi qonunda o’quv jarayoniga ilg’or pedagogik texnologiyalarni joriy etish mamlakatimiz ta’lim tizimini isloh qilishning asosiy ko’rsatkichlaridan biri sifatida e’tirof etilishi bejiz emas. Chunki, pedagogik texnologiya ta’lim jarayonini inqirozdan holi etish, uni bozor iqtisodi sharoitiga mos holda takomillashtirish va Davlat ta’lim standarti talablariga muvofiq kadrlar tayyorlashning muhim omillaridan biri bo’lib hisoblanadi.

Ta’lim tizimini zamonaviy axborot va pedagogik texnologiyalari bilan jihozlash, o’quv jarayonida zamonaviy kompyuterlar imkoniyatlaridan samarali foydalanish, ta’lim sifatini oshirishda musobaqa, olimpiadalar o’tkazish, o’quvchilarning fikrlash qobiliyati, intellektual salohiyatini oshirish asosiy vazifalardan biri bo’lmoqda.

Zero ta’lim jarayonini tashkillashda axborot–kommunikatsiya texnologiyalaridan samarali foydalanayotgan ko’pgina rivojlangan mamlakatlar o’quv jarayonini sifat jihatdan yangi bosqichga ko’tarishga erishayotganliklari hayotda o’z tasdig’ini topmoqda. Ushbu bitiruv malakaviy ishda bayon etilgan ma’lumotlar esa Dasturlashga oid olimpiada masalalari va ularni yechish metodikasi usullaridan samarali foydalanish salohiyati to’g’risida fikr yuritishga asos bo’ladi.

Yuqorida keltirilgan ta’riflarni analiz qilish asosida quyidagi xulosani chiqarish mumkin: Informatikadan o’quv masalasi dasturning qonun va metodlar asosida o’quvchilardan fikrlashni va amaliy faoliyatni talab qiladi. Bunday masalalarni yechishda, algoritm, dasturlash bilimlarni egallashda, ularni amalda qo’llanish uquvini hosil qilishga va fikrlashni rivojlantirishga asosiy e’tibor beriladi. O’quv masalasi o’qitish metodi sifatida turli funksiyalarni, ya’ni bilim berish, tarbiyalash, aqliy rivojlantirish, o’qitishni tashkil qilish, bilimlarni tekshirish kabi funksiyalarni bajaradi. O’quv masalalarining bunday turli

funksiyalari masala yechish metodlarini va usullarini egallashda uning imkoniyatlarini aniqlaydi

Mening fikrimcha ta’limda ko’plab innovatsion usullar samarali bo’lib, ayniqsa ta’limda aniq maqsad qo’yish yoki fanlararo bog’lash, aniq maqsadga erishish ko’p foydali xisoblanadi. «Maqsadli yondashuv» uslubi ta’limda o’quvchi, talabalarni ijodiy fikrlashga, fanning mazmun mohiyati va maqsadini o’rgatadi. U nafaqat o’quv-bilim faoliyatini tizimlashtiradi, jadallashtiradi, balki ta’lim oluvchilarning umumiy madaniyatini yuksaltirishga ham xizmat qiladi, predmetlar aro bog’lanish tufayli maqsadlar turli darajada namoyon bo’ladi.

Masalalar amaliy va o’quv masalalariga ajraladi. O’quv masalalari o’quv faoliyatining asosini tashkil qiladi. Amaliy masalalar amalga oshirilganda olinadigan natijada masala yechish jarayoni qatnashmaydi, o’quv masalasi amalga oshirilganda esa masala yechish jarayoni qatnashadi, chunki u bevosita natija, bo’lib hisoblanadi. O’quv jarayonida olimpiada masalasi deb, odatda unchalik katta bo’lmagan biror konkret muammoga aytiladi. Bunday masalalar mantiqiy fikr yuritish, matematik amallarni bajarish va dasturlash tillarining qonunlaridan hamda masalalarni yechish metodlaridan foydalanib, eksperiment o’tkazish asosida yechiladi.

Bitiruv malakaviy ishning maqsadi maktab, akademik litsey, kasb-xunar kollejlari, universitetlarida bo’lajak dasturchi mutaxassislarni tayyorlashda ta’limning faol usullari va ulardan foydalanish masalalarini yaratishdan iborat.

Bu maqsad va vazifalarga erishish uchun bu sohaga doir ilmiy va metodik adabiyotlar batafsil o’rganildi.

Natijada o’quvchilarga ta’lim berishning faol usullaridan foydalanish bo’yicha uslubiy ko’rsatmalar yaratishga erishildi.

Ushbu bitiruv malakaviy ishda yoritilgan fikr va mulohazalardan foydalanish nitijasida har bir o’quvchi quyidagi natijalarga erishadi:

-fanlardan tashkil qilinadigan masalalarni tahlil qilish va ularning algoritmini tuzish va modellashtirish;

-Masalani yechish usuli (algoritmi) ni tanlash va tasvirlash;

-Yaratilgan algoritmni dasturlash tiliga o’tkazish va kamchiliklar va xatoliklarini bartaraf qilish va dasturni takomillashtirish;

-O’quvchi yoshlarda dasturlash malakasini shakllantirish;

Bitiruv malakaviy ishida keltirilgan ma’lumotlardan AKT soxasidagi yosh mutaxassislar qo‘llanma sifatida foydalanishlari mumkin, undan tashqari ta’lim jarayonida “Algoritm va dasturlash” kabi fanlarning dars mashg‘ulotlarida keng foydalanishlari xamda talabalar ushbu fanlar bo‘yicha o‘zlarining nazariy bilimlari va amaliy ko‘nikmalarini shakllantirishlari mumkin.

Ushbu bitiruv malakaviy ishini bajarish va tayyorlashda o‘z mutaxassisligimga oid bir qancha amaliy bilim va ko‘nikmalarga ega bo‘ldim va bu bilim va ko‘nikmalarni kelajakda ishlab chiqarishda va ta’lim soxasida qo‘llashga harakat qilaman.

FOYDALANILGAN ADABIYOTLAR RO'YHATI

1. O'zbekiston Respublikasining "Ta'lim to'g'risida" gi 1997-yil 29-avgustdagi 464-I-sonli qonuni.

2. O'zbekiston Respublikasining "Kadrlar tayyorlash milliy dasturi", 1997- yil.

3. O'zbekiston Respublikasining "Axborotlashtirish to'g'risida" gi 2003-yil 11-dekabrdagi qonuni.

4. O'zbekiston Respublikasi Prezidenti Mirziyoev Sh.M. ning PF-4947 sonli Farmoni. O'zbekiston Respublikasini yanada rivojlantirishning beshta ustuvor yo'nalishi bo'yicha "Xarakatlar strategiyasi". T.: 2017 yil

5. « Dasturlash » Sh.I.Razzoqov, M.J.Yunusova - kasb-hunar kollejlari uchun o'quv qo'llanma. Toshkent – « ILM ZIYO » - 2006 y.

6. « Dasturlash bo'yicha praktikum » M.J.Yunusova, A.B.Rahimov. kasb-hunar kollejlari uchun o'quv qo'llanma. Toshkent – « ILM ZIYO » - 2006 y.

7. « Informatika va hisoblash texnikasi asoslari » umumiy o'rta ta'lim maktablarining 9-sinfi uchun darslik, 2-nashri. B.J.Boltayev, A.R.Azamatov, A.D.Asqarov va boshqalar. Cho'lpon nomidagi nashriyot - matbaa ijodiy uyi, Toshkent – 2015.

8. G.Grigas. Dasturlash asoslari. T., «O'qituvchi», 1990.

9. М.Зелковиц. А.Шоу, Дж.Гэннон. Принципы разработки программного обеспечения. Пер. с англ. М., «Мир», 1982.

10. Дж. Фокс. Программное обеспечение и эго разработка. Пер. с англ. М., «Мир», 1985.

11. Н. Вирт. Алгоритмы и структуры данных. Пер. с англ. М., «Мир», 1989.

12. А.Л. Брудно, Л.И.Каплан. Московские олимпиады по программированию. 2-е издание. М., «Наука», 1990.

13. В.Ф. Очков, Ю.Ю.Пухначев. 128 советов начинающему программисту. М., 1991.

14. А.В. Файсман. Профессиональное программирование на Турбо Паскале. Т., Информ Экс – корпорейшн, 1992.

15. В.В. Фаронов. Turbo Pascal 7.0. М., Нолидж, 2000.

16. Н.Б. Культин. Turbo Pascal в задачах и примерах. СПб, БХВ - Санкт-Петербург, 2001

17. А.Н. Марченко. Программирование в среде Turbo Pascal 7.0. К., Век+, М., ДЕСС, 1999.

18. « Algoritmlar nazariyasi » fani bo'yicha ma'ruzalar matni. Axadov Akmal Rustamovich. Samarqand-2011.

Internet saytlari:

1. www.lex.uz

2. www.ziyonet.uz

3. www.acm.tuit.ru

4. www.tami.uz

5. www.e-dastur.uz