INNOVATIVE PUBLICATION
Journal of Effective Learning and Sustainable Innovation
Vol.2 №2 (2024). February
innovativepublication.uz

TALABALARGA DASTURLASH JARAYONLARINI INTEGRATIV YONDOSHUV ASOSIDA O‘RGATISH

Himmatov Shohrux Oripovich
Samarqand davlat universitetining
Kattaqo‘rg‘on filiali Raqamli
texnologiyalar va iqtisodiyot
kafedrasi o‘qituvchisi

Annotatsiya: Bugungi kunda universitet va maktablarda dasturlashni o’qitish metodikasini takomillashtirish muammosi dolzarb bo‘lib qolmoqda. Talabalarning yangi o‘qitish uslub va shakllari zarurligini inobatga olib dasturlashni integrativ yondoshuv asosida o‘rgatish tavsiya etiladi. Muallif dasturlashni o‘rgatish bo‘yicha o‘z tajribasini taqdim etadi va ilg‘or texnologiyalarni yoyish maqsadida dasturlashni o‘rgatishda chiziqli jaroyonlarni, tarmoqlanuvchi jarayonlarni hamda takrorlanuvchi jarayonlarni hayotiy misollar bilan tushuntirishni taklif qiladi.

Kalit so’zlar: chiziqli jarayonlar, tarmoqlanuvchi jarayonlar, takrorlanuvchi jarayonlar, massivlar, hayotiy misollar.

Chiziqli dasturlash - bu, qarorlar qabul qilish bilan bog‘liq muammolarni hal qilish usuli va u ko‘plab hayotiy vaziyatlarda foydali bo‘lishi mumkin. Keling, buni sizga ba'zi misollar bilan tushuntiraman.

Tasavvur qiling, siz tug‘ilgan kuningizni rejalashtirmoqdasiz va sizning byudjetingiz cheklangan. Siz tug‘ilgan kuningizga pitsa yoki tort buyurtma qilmoqchisiz. Qancha pitsa va qancha tortga buyurtma berishni aniqlamoqchisiz. Tug‘ilgan kuningizdagi mehmonlarning barchasini xursand qilishingiz va shu bilan birga mavjud byudjetingiz doirasida ish qilishingiz kerak. Bunda chiziqli dasturlash yordam beradi!

Aytaylik, pitsa 10$, tort esa 20$ turadi. Siz tug‘ilgan kuningiz uchun maksimal 100$ sarflashingiz mumkinligini bilasiz. Endi siz byudjetni oshirmasdan ko‘pchilikni xursand qilish uchun qancha pitsa va tortlarga buyurtma berish kerakligini aniqlab olishingiz kerak.

Bu muammoni tenglamalar yordamida ifodalashimiz mumkin. Keling, pitsalar sonini "P" va tortlar sonini "T" deb ataymiz. Muammoga ko‘ra, biz bilamizki, pitsa narxi (10P) va tortlarning narxi (20T) 100 dollar(sizning byudjetingiz)dan kam yoki unga teng bo‘lishi kerak .

Aytaylik, har bir pitsa 5 kishiga, har bir tort esa 8 kishga yetadi. Siz ziyofatda hamma uchun yetarli pitsa yoki tort borligiga ishonch hosil qilishni xohlaysiz. Shunday qilib, ziyofatdagi odamlar soni pitsa bilan oziqlanadigan odamlar sonidan (5P) va tort bilan oziqlangan odamlar sonidan (8T) kam yoki teng bo‘lishi kerak.

Va nihoyat, biz imkon qadar ko‘proq odamlarni xursand qilishni xohlaymiz. Shunday qilib, bizning maqsadimiz 5P + 8T tenglamasi bilan ifodalanadigan tug‘ilgan kundagi odamlar sonini maksimal darajada oshirishdir.

Ushbu tenglamalarni chiziqli dasturlash usullaridan foydalangan holda yechish orqali biz byudjetimiz doirasida ko‘pchilikni xursand qiladigan pitsa va tortlarning eng yaxshi kombinatsiyasini topishimiz mumkin.

Bu misolda, deylik, 4 ta pitsa va 3 ta tortga buyurtma berish bizning byudjetimizdan oshib ketmasdan ko‘pchilikni xursand qiladi. Shunday qilib, tug‘ilgan kunni rejalashtiruvchi sifatida siz qiziqarli va arzon ziyofat o‘tkazish uchun 4 ta pitsa va 3 ta tortga buyurtma berishga qaror qilasiz.

Chiziqli dasturlashdan boshqa real vaziyatlarda ham foydalanish mumkin. Misol uchun, do‘kon o‘z sotuvlarini maksimal darajada oshirish uchun qancha mahsulotni saqlash kerakligini hal qilish uchun undan foydalanishi mumkin. Yoki talaba undan eng yaxshi baho olish uchun turli fanlarni o‘rganishga qancha soat sarflashini hal qilish uchun foydalanishi mumkin.

Shunday qilib, chiziqli dasturlash bizga cheklangan resurslar yoki erishish uchun aniq maqsadlar mavjud bo‘lganda qarorlar qabul qilish va eng yaxshi yechimlarni topishga yordam beradigan foydali vosita bo‘la oladi. Bu eng yaxshi javobni topish uchun jumboqni yechishga o‘xshaydi!

Python tilida masalani yechish uchun quyidagi dasturni ishlatish mumkin:

```
from pulp import *
# Maqsadimiz 5P + 8T ni maksimal oshirish
problem = LpProblem("Pitsa va Tort", LpMaximize)
# Variablarni aniqlash
P = LpVariable("P", lowBound=0, cat='Integer')
T = LpVariable("T", lowBound=0, cat='Integer')
# Maqsadni qo‘llash
problem += 5 * P + 8 * T
# Cheklashlar (shartlar)
problem += 10 * P + 20 * T <= 100
problem += 5 * P <= 8 * T
# Muammoga yechim topish
problem.solve()
# Yechimni chop etish
print("Optimal kombinatsiya:")
print("Pitsalar soni:", value(P))
print("Tortlar soni:", value(T))
```

Yuqoridagi dasturda, problem degan LpProblem obyekti yaratiladi va "Pitsa va Tort" nomi bilan maqsad aniqlanadi. Keyin P va T degan LpVariable obyektlari aniqlanadi, ular pitsalar va tortlar sonini ifodalaydi.

Problem obyektiga maqsad 5 * P + 8 * T ni maksimal oshirish deb qo‘shiladi. Shartlar problem += 10 * P + 20 * T <= 100 (byudjetga cheklangan) va problem += 5 * P <= 8 * T (ziyofatga oid cheklangan) deb qo‘shiladi.

Keyin problem.solve() ni chaqirib, masalani yechish jarayoniga o‘tkazamiz. Yechim topilgandan so‘ng, P va T ning qiymatlari value(P) va value(T) yordamida chiqariladi.

Yuqorida berilgan misolda, byudjet doirasida 4 ta pitsa va 3 ta tort buyurtma berishning eng yaxshi kombinatsiyasi hisoblanadi. Siz shunday qilib tug‘ilgan kuningizdagi odamlarni ko‘proq xursand qilasiz, ayniqsa byudjet cheklanganligi asosida.

Tarmoqlanuvchi dasturlar tanlov qilish va turli yo‘llardan borish orqali muammolarni hal qilish usulidir. Ular ko‘plab real vaziyatlarda juda qiziqarli va foydali bo‘lishi mumkin. Keling, buni sizga ba'zi misollar bilan tushuntiraman.

Tasavvur qiling, siz xazina oviga ketyapsiz. Sizda xazinani topish uchun turli yo‘llarni ko‘rsatadigan xaritangiz bor. Har bir yo‘lda turli xil to‘siqlar va maslahatlar mavjud. Bu erda tarmoqlanuvchi dasturlar yordam berishi mumkin!

Siz xaritaning boshidan boshlaysiz va tanlov qilishingiz kerak. Aytaylik, birinchi tanlov chapga yoki o‘ngga borishdir. Siz tanlagan tanlovingizga qarab, qaysidir yo‘ldan borasiz va turli to‘siqlar va maslahatlarga duch kelasiz.

Misol uchun, agar siz chapga borishni tanlasangiz, siz chiqishingiz kerak bo‘lgan daraxtga yoki hal qilishingiz kerak bo‘lgan jumboqga duch kelishingiz mumkin. Ammo agar siz o‘ngga borishni tanlasangiz, siz kesib o‘tishingiz kerak bo‘lgan daryo yoki borishingiz kerak bo‘lgan labirintni topishingiz mumkin.

Xazina ovining har bir nuqtasida siz qaror qabul qilishingiz va ma'lum bir yo‘lni bosib o‘tishingiz kerak. Har bir qaror boshqa natijaga olib keladi va sizni xazinani topishga yaqinlashtiradi yoki uzoqlashtiradi.

Tarmoqli dasturlardan boshqa real vaziyatlarda ham foydalanish mumkin. Misol uchun, do‘stlaringiz bilan qanday o‘yin o‘ynashni hal qilayotganingizda, sizda futbol, basketbol yoki valeybol o‘ynash kabi turli xil variantlar bo‘lishi mumkin. Har bir variant boshqa o‘yin va turli qiziqarli tajribalarga olib keladi.

Yoki oilangiz bilan dam olishni rejalashtirganingizda, plyajga yoki tog‘larga borishni tanlashingiz kerak bo‘lishi mumkin. Har bir tanlov boshqa manzilga va turli sarguzashtlarga olib keladi.

Tasavvur qilaylik, siz muzqaymoq sotib olmoqchisiz. Sizda 10 $ bor va siz o‘zingizning sevimli shokoladingizni sotib olishingiz mumkinmi yoki boshqa lazzatni tanlashingiz kerakmi yoki yo‘qligini bilishni xohlaysiz. Shartli operatorlar qanday yordam berishi mumkin:

1. Agar shokoladli muzqaymoq narxi 10 $ dan kam yoki teng bo‘lsa, uni sotib olishingiz mumkin. Ammo agar u 10 $dan ortiq bo‘lsa, unda siz boshqa mahsulotni tanlashingiz kerak.
2. Aytaylik, shokoladli muzqaymoq narxi 5 $. 5$ 10$ dan kam bo‘lganligi sababli, shart to‘g‘ri va siz o‘zingizning sevimli muzqaymog‘ingizni sotib olishingiz mumkin!
3. Ammo shokoladli muzqaymoqning narxi 12 $ bo‘lsa, bu 10 $ dan ortiq bo‘lsa, shart noto‘g‘ri. Bunday holda, siz byudjetingizga mos keladigan boshqa mahsulatni tanlashingiz kerak bo‘ladi.

Aniqroq tushuntirish uchun quyidagi shartlar bilan bir dastur yozamiz:

```
budget = 10 # Byudjetimiz
chocolate_price = 12 # Shokoladli muzqaymoq narxi
if chocolate_price <= 10:
    print("Siz o‘zingizning sevimli shokoladingizni sotib olishingiz mumkin!")
elif chocolate_price <= 5:
    print("Siz o‘zingizning sevimli muzqaymoqingizni sotib olasiz!")
else:
    print("Sizning byudjetingizga mos keladigan boshqa mahsulotni tanlashingiz kerak.")
```

Yuqoridagi dasturda, budget degan o‘zgaruvchi bo‘yicha byudjetimizni va chocolate_price degan o‘zgaruvchi bilan shokoladli muzqaymoqning narxini ifodalaymiz.

Agar chocolate_price 10 dan kam yoki teng bo‘lsa, if qismidagi shart to‘g‘ri bo‘ladi va "Siz o‘zingizning sevimli shokoladingizni sotib olishingiz mumkin!" deb ekranga chiqadi.

Agar chocolate_price 5 dan kam bo‘lsa, ammo 10 dan ortiq bo‘lmasa, elif qismidagi shart to‘g‘ri bo‘ladi va "Siz o‘zingizning sevimli muzqaymoqingizni sotib olasiz!" deb ekranga chiqadi.

Aks holda, yani chocolate_price 10 dan katta bo‘lsa, else qismidagi shart bajariladi va "Sizning byudjetingizga mos keladigan boshqa mahsulotni tanlashingiz kerak." deb ekranga chiqadi.

Shu dastur orqali siz muzqaymoqning narxini bilib, byudjetingizga qarab shokoladli muzqaymoqni sotib olishingiz kerakligini aniqlashingiz mumkin. Shu bilan birga, shartli operatorlarning qanday yordam berishi va ulardan foydalanish mumkinligini tushuntirishga harakat qildik.

Shartli operatorlardan boshqa real vaziyatlarda ham foydalanish mumkin. Keling, soyabon olib kelish yoki kelmaslikni hal qilishning misolini ko‘rib chiqaylik:

1. Agar ob-havo ma'lumotlariga ko‘ra, bugun yomg‘ir yog‘ishi aytilgan bo‘lsa, unda siz soyabon olib kelishingiz kerak. Ammo agar yomg‘ir yog‘maydi, deb aytsa, unda siz olib kelishingiz shart emas.

2. Aytaylik, ob-havo ma'lumoti yomg‘ir yog‘ishini taxmin qilmoqda. Vaziyat to‘g‘ri bo‘lgani uchun, quruq qolish uchun soyabonni olib kelishingiz kerak.

3. Ammo ob-havo prognozi yomg‘ir yog‘masligini taxmin qilsa, shart noto‘g‘ri va siz soyabonni uyda qoldirishingiz mumkin.

Ikkala misolda ham shartli operatorlar ma'lum shartlar asosida qaror qabul qilishimizga yordam beradi. Ular bizga real hayotda sodir bo‘layotgan voqealarga qarab turli xil harakatlarni tanlash imkonini beradi.

Shunday qilib, xuddi xazina ovida tanlov qilish yoki qanday o‘yinni o‘ynashni tanlash kabi, dasturlardagi shartli operatorlar bizga real hayotiy vaziyatlarda tanlov qilishimizga yordam beradi. Ular "agar", "boshqa" va "keyin" kabi belgilardan foydalanib, bizni duch keladigan sharoitlardan kelib chiqqan holda eng yaxshi qarorga yo‘naltiradi.

Takrorlanuvchi dasturlar - kompyuter dasturlarining alohida qismi bo‘lib, bizga ma'lum harakatlarni qayta-qayta takrorlashga yordam beradi. Ular ma'lum bir shart bajarilmaguncha davom etadigan tugamaydigan siklga o‘xshaydi.

Tasavvur qilaylik, siz imkon qadar ko‘proq tanga to‘plashingiz kerak bo‘lgan o‘yin o‘ynayapsiz. Dastlab sizda tangalar sonini nol bilan boshlaysiz va ma'lum bir raqamga, aytaylik 10 ga yetguncha ularni yig‘ishni davom ettirmoqchisiz. Takrorlanuvchi dasturlar qanday yordam berishi mumkin:

1. "Tangalar soni 10 dan kam bo‘lsa, ko‘proq tangalar yig‘ishni davom eting" degan takrorlanuvchi dasturini yaratishingiz mumkin.
2. Shunday qilib, siz o‘yin o‘ynashni boshlaysiz va bitta tanga yig‘asiz. takrorlanuvchi dastur tangalar soni 10 dan kam yoki yo‘qligini tekshiradi va tangalar soni 10 taga yetmaguncha ishni davom ettirish imkonini beradi.
3. Siz yana bir tanga yig‘asiz, jami ikkitasini qilasiz. Takrorlanuvchi dastur yana tekshiradi va shart hali ham to‘g‘ri ekanligini ko‘radi, shuning uchun siz yig‘ishni davom ettirishingiz mumkin.
4. Ushbu jarayon 10 ta tanga yig‘guningizcha davom etadi. Bu vaqtda takrorlanuvchi dastur shartni yana tekshiradi va tangalar soni 10 dan kam emasligini ko‘radi. Shunday qilib, u ishni to‘xtatadi va siz maqsadingizga erishdingiz!

Takrorlanuvchi dasturlash o‘zgaruvchilar, shartlar va takrorlanishdan iborat bo‘lishi mumkin. Sizning o‘rnatingizdagi dastur ushbu jarayonni amalga oshiradi. Siz uchun misol:

```
tangalar_soni = 0
while tangalar_soni < 10:
    tangalar_soni = tangalar_soni + 1
print("Jami tangalar soni:", tangalar_soni)
```

Ushbu dastur takrorlanayotgan sikl yordamida tangalar_soni o‘zgaruvchisini oshirib boradi, va sikl tugaganida jami tangalar sonini chiqaradi. Bu dastur jami 10 ta tanga yig‘ayotganizda ishni tugatadi.

Siz bu dasturning o‘zgartirishlarni amalga oshirib, boshqalar bilan ham o‘ynash imkoniyatiga ega bo‘lasiz. Misol uchun, agar bir do‘stingiz bilan o‘ynayotgan bo‘lsangiz, siz har bir jangni o‘ynashda tangalar_soni ni qayta nolga olib boshlashingiz mumkin:

```
tangalar_soni
= 0
while tangalar_soni < 10:
    tangalar_soni = tangalar_soni + 1
  print("Jami tangalar soni:", tangalar_soni)
print("O‘yin tugadi!")
```

Ushbu dastur sizga har bir jangda jami tangalar sonini ko‘rsatadi va o‘yin tugadiganingizda "O‘yin tugadi!" degan xabar chiqaradi.

Takrorlanuvchi dasturlardan boshqa real vaziyatlarda ham foydalanish mumkin. Keling, pechene pishirish misolini ko‘rib chiqaylik:

1. Tasavvur qiling, siz pechenye pishirasiz va ularni 15 daqiqa davomida pishirishingiz kerak. Siz "15 daqiqa o‘tmaguncha pechanyeni pishirishda davom eting" degan takrorlanuvchi dasturni yaratishingiz mumkin.

2. Pechanyelarni pechga qo‘yasiz va taymerni ishga tushirasiz. Takrorlash operatori 15 daqiqa o‘tganligini tekshiradi, agar 15 daqiqa o‘tmagan bo‘lsa, pishirishni davom ettirishga imkon beradi.

3. Vaqt o‘tishi bilan takrorlanuvchi dastur 15 daqiqa o‘tganligini tekshiradi. Agar 15 daqiqa o‘tgan bo‘lsa shu nuqtada dastur to‘xtaydi.

Quyidagi dastur sizning maqsadingizni amalga oshiradi:

```
import time
pishirish_vaqt = 0
while pishirish_vaqt < 15:
    print("Pechenyeleri pishirish davom etmoqda...")
    time.sleep(1) # 1 sekund kutamiz
    pishirish_vaqt += 1
print("Pechenyeler pishirildi!")
```

Ushbu dasturda time.sleep(1) funksiyasi yordamida dastur bir sekund kutadi, keyin esa pishirish_vaqt o‘zgaruvchisini 1 ga oshiradi. Bunda pishirish_vaqt o‘zgaruvchisi 15 ga teng bo‘lgan paytgacha siklni takrorlaydi.

Dastur yurishi yaqinda o‘zgartirilganidan so‘ng, "Pechenyeleri pishirish davom etmoqda..." degan xabarlarni har sekundda chiqaradi. 15 daqiqa o‘tgandan so‘ng esa "Pechenyeler pishirildi!" degan xabar chiqaradi va dastur tugaydi.

Ikkala misolda ham takrorlanuvchi dasturlar ma'lum bir shart bajarilmaguncha ma'lum harakatlarni takrorlashga yordam beradi. Ular bizga maqsadimizga erishgunimizcha yoki biror vazifani bajarmagunimizcha davom etishimizga imkon beradi.

Shunday qilib, xuddi o‘yinda tanga yig‘ish yoki pechenyelarni pishirish kabi, kompyuter dasturlaridagi takrorlanish dasturlari bizga real hayotdagi harakatlarni

takrorlashga yordam beradi. Ular biz o‘z oldimizga qo‘ygan narsaga erishgunimizcha davom etishimizni ta'minlaydi.

Dasturlashdagi **massivlar** bizga ko‘p ma'lumotlarni saqlash va tartibga solishda yordam beradigan maxsus konteynerlarga o‘xshaydi. Ular har xil bo‘linmalari bo‘lgan katta qutiga o‘xshaydi, bu erda biz turli narsalarni qo‘yishimiz mumkin.

Tasavvur qilaylik, sizda sevimli o‘yinchoqlar to‘plami bor va siz ularni tartibli saqlashni xoxlaysiz. Bunda massivlar sizga yordam berishi mumkin:

1. Siz "toyCollection" deb nomlangan massiv yaratasiz va uning ichiga barcha o‘yinchoqlaringizni joylashtirasiz. Har bir o‘yinchoq massivdagi alohida bo‘limga yoki uyaga joylashtiriladi.
2. Aytaylik, sizda 5 ta o‘yinchoq bor: ayiqcha, qo‘g‘irchoq, mashina, boshqotirma va to‘p. Siz har bir o‘yinchoqni massivdagi boshqa uyaga qo‘yasiz, masalan birinchi uyada ayiqcha, ikkinchi uyada qo‘g‘irchoq va hokazo.
3. Endi, agar siz mashinangizni topmoqchi bo‘lsangiz, uni oddiygina qatordan izlashingiz mumkin. Siz uni uchinchi uyaga qo‘yganingiz uchun, bilasizki, sizning mashinangiz o‘yinchoq massivning uchinchi bo‘linmasida.
4. Massivlar ham hisob-kitoblarni amalga oshirishda yordam beradi. Masalan, har bir o‘yinchoqning o‘ziga xos qiymati bor deylik. Masalan: ayiqcha 5 ball, qo‘g‘irchoq 7 ball, mashina 10 ball, boshqotirma 3 ball, to‘p 2 ballga baholanadi.
5. Agar siz o‘yinchoq kolleksiyangizning umumiy qiymatini bilmoqchi bo‘lsangiz, ushbu qiymatlarni saqlash uchun massivdan ham foydalanishingiz mumkin. Massivdagi har bir o‘yinchoqning qiymatlarini qo‘shib, o‘yinchoqlar kollektsiyangizning umumiy qiymati 27 ball ekanligini hisoblashingiz mumkin.

Quyidagi dastur misoli bu jarayonni namoyish etadi:

```
toyCollection = ["ayiqcha", "qo‘g‘irchoq", "mashina", "boshqotirma", "to‘p"]

# Mashina uchun indeksni topish
mashina_indeks = toyCollection.index("mashina")
print("Mashina indeksi:", mashina_indeks)
# O‘yinchoqlar qiymatlarini belgilash
o’yinchoq_qiymatlari = [5, 7, 10, 3, 2]
# Umumiy qiymatni hisoblash
umumiy_qiymat = sum(o’yinchoq_qiymatlari)
print("O‘yinchoqlar kollektsiyasining umumiy qiymati:", umumiy_qiymat)
```

Ushbu dasturda toyCollection nomli tartiblangan massiv yaratiladi va har bir o‘yinchoqning nomi joylashtiriladi. mashina_indeks o‘zgaruvchisi, toyCollection massivida "mashina" nomli o‘yinchoqning indeksini topadi.

o’yinchoq_qiymatlari nomli massiv yaratiladi va har bir o‘yinchoq uchun o‘ziga xos qiymatlar beriladi. Misol uchun, o’yinchoq_qiymatlari massivi ayiqcha uchun 5, qo‘g‘irchoq uchun 7, mashina uchun 10, boshqotirma uchun 3 va to‘p uchun 2 qiymatlarini o‘z ichiga oladi.

umumiy_qiymat o‘zgaruvchisi sum() funksiyasi yordamida o’yinchoq_qiymatlari massivdagi qiymatlar yig‘indisini hisoblaydi va umumiy o‘yinchoqlar kollektsiyasining qiymatini chiqaradi.

Siz ham massivlar yordamida sevimli o‘yinchoqlaringizni tartiblangan holda saqlayishingiz mumkin. Ularning indeksini topish, qiymatlarini belgilash va umumiy qiymatni hisoblash imkoniyatiga ega bo‘lasiz.

Massivlar ko‘plab real vaziyatlarda qo‘llanilishi mumkin. Misol uchun, sizda sinfingizda 10 nafar talaba bor va siz ularning test ballarini kuzatib borishni xohlaysiz deb

tasavvur qiling. Siz “testScores” deb nomlangan massiv yaratishingiz va har bir talabaning ballini massivdagi uyaga belgilashingiz mumkin. Keyin, siz ularning ballarini osongina topishingiz va solishtirishingiz mumkin.

Xulosa qilib aytadigan bo‘lsak, dasturlashdagi massivlar bizga ko‘plab ma'lumotlarni saqlash va tartibga solishga yordam beradigan konteynerlarga o‘xshaydi. Ular bizga turli narsalarni kuzatib borish, hisob-kitoblarni amalga oshirish va aniq narsalarni osongina topish imkonini beradi. O‘yinchoqlaringizni tartibga solish yoki test natijalarini kuzatish kabi massivlar bizga kompyuter dasturlaridagi ma'lumotlarni boshqarish va ularga kirishda yordam beradi.

Foydalanilgan adabiyotlar ro’yxati

1. *Dann V., Mis S., Pausch R. Elis bilan dasturlashni o‘rganish. Upper Saddle River, NJ: Prentice Hall, 2006 yil.*
2. *Kelleher C., Pausch R. Dasturlash uchun to‘siqlami kamaytirish: Ajarn dasturchilar uehun dasturlash muhiti va tillarining taksonomiyasi. ACM Computing Surveys, 37(2), 83-137- bctlar, 2008 yil 28 mart.*
3. Tokhirov F. J. Algorithmic Thinking of Students in Program using Electronic Learning Resources Principles in Development //Kresna Social Science and Humanities Research. – 2022. – T. 3. – C. 93-94.