INNOVATIVE PUBLICATION

Journal of Effective Learning and Sustainable Innovation

Vol.3 №5 (2025). May

innovativepublication.uz

C# DASTURLASH TILIDA KO'P O'LCHOVLI VA NOTEKIS MASSIVLAR BILAN ISHLASH

Islomova Tamanno Ikrom qizi

Farg‘ona davlat universiteti talabasi

islomovatamanno4@gmail.com

Mirsaid Yusupov Abdulazizovich

mirsaidbeky@gmail.com

Annotatsiya: Mazkur maqolada C# dasturlash tilida ko‘p o‘lchovli va notekis (jagged) massivlar bilan ishlashning nazariy va amaliy jihatlari yoritilgan. Massivlarning tuzilishi, ularning farqlari, afzalliklari va dasturlarda qo‘llanilishi tahlil qilingan. Shuningdek, massivlar bilan ishlashda uchraydigan texnik muammolar va ularning yechimlari haqida ham fikr yuritilgan. Maqola dasturlash asoslarini o‘rganayotgan talabalar va dasturchilar uchun foydalidir.

Kalit so‘zlar: C# dasturlash tili, ko‘p o‘lchovli massiv, notekis massiv, indekslash, xotira tashkiloti, massivlar farqi.

Аннотация: В данной статье рассматриваются теоретические и практические аспекты работы с многомерными и зубчатыми (jagged) массивами в языке программирования C#. Проанализированы структура массивов, их различия, преимущества и области применения в программировании. Также затронуты технические сложности при работе с массивами и возможные способы их решения. Статья будет полезна студентам и программистам, изучающим основы программирования.

Ключевые слова: язык программирования C#, многомерный массив, зубчатый массив, индексация, организация памяти, различие массивов

Abstract: This article explores the theoretical and practical aspects of working with multidimensional and jagged arrays in the C# programming language. It analyzes the structure of arrays, their differences, advantages, and application in real-world programming scenarios. Technical issues encountered when dealing with arrays and their solutions are also discussed. The article is beneficial for students and developers studying programming fundamentals.

Keywords: C# programming language, multidimensional array, jagged array, indexing, memory organization, array differences.

Massivlar zamonaviy dasturlash tillarida muhim ma’lumot tuzilmalari hisoblanadi. Ular bir xil turdagi elementlardan tashkil topgan bo‘lib, dasturchiga katta hajmdagi ma’lumotlarni tartibli va qulay tarzda saqlash imkonini beradi. Ayniqsa, C# dasturlash tilida massivlar orqali bir nechta o‘lchamli ma’lumotlar bilan ishlash imkoniyati mavjud. Ushbu maqolada aynan ko‘p o‘lchovli (multidimensional) va notekis (jagged) massivlar bilan ishlash, ularning o‘ziga xos xususiyatlari, farqlari va amaliy dasturlarda qanday qo‘llanilishi haqida batafsil fikr yuritiladi.

Avvalo, ko‘p o‘lchovli massivlar haqida gapirilganda, bu ikki yoki undan ortiq o‘lchovga ega bo‘lgan massivlar tushuniladi. Oddiy massivlar faqat bitta o‘lchovga ega bo‘lsa (ya’ni, ular bir qatordagi elementlar ketma-ketligidir), ko‘p o‘lchovli massivlar esa ikki o‘lchovli (matrisa ko‘rinishidagi), uch o‘lchovli yoki undan ortiq bo‘lishi mumkin. Ko‘p o‘lchovli massivlarning eng keng tarqalgan shakli ikki o‘lchovli massivdir, u odatda satr va ustunlar orqali tasvirlanadi.

C# dasturlash tilida ikki o‘lchovli massiv quyidagicha e’lon qilinadi:

int[,] sonlar = new int[3, 4];

Yuqoridagi misolda sonlar nomli massiv 3 ta satr va 4 ta ustundan iborat bo‘lib, jami 12 ta butun sonni o‘zida saqlashi mumkin. Ushbu massivlar kompyuter xotirasida satrma-satr yoki ustunma-ustun shaklida joylashadi (C# tilida satrma-satr holatda), bu esa ularning ish faoliyatiga ta’sir qilishi mumkin.

Ko‘p o‘lchovli massivlarning afzalligi shundaki, ular murakkab strukturalashtirilgan ma’lumotlar bilan ishlashda qulaylik yaratadi. Masalan, jadval ko‘rinishidagi ma’lumotlarni (talabalar baholari, sport musobaqalari natijalari, temperaturalar jadvali va h.k.) saqlashda ikki o‘lchovli massivlar ideal yechim bo‘la oladi. Bunday massivlarda ma’lumotlar satr va ustun koordinatalari orqali aniqlanadi.

Ko‘p o‘lchovli massivlardan foydalanishda indekslar yordamida kerakli elementga murojaat qilinadi. Masalan, sonlar[2, 1] ifodasi uchinchi satr va ikkinchi ustundagi elementga murojaat qiladi. Indekslar 0 dan boshlanishini eslatib o‘tish muhimdir.

Ko‘p o‘lchovli massivlar C# dasturlash tilida ma’lumotlarni tartibli saqlash va qayta ishlashda muhim o‘rin tutadi. Ular turli sohalardagi dasturlarda, jumladan, ilmiy hisoblashlar, grafik ilovalar, matritsa amallari va boshqa ko‘plab amaliyotlarda keng qo‘llaniladi.

Ko‘p o‘lchovli massivlardan tashqari, C# dasturlash tilida yana bir muhim ma’lumot tuzilmasi mavjud bo‘lib, bu **notekis massivlar** (inglizcha nomi: *jagged arrays*) deb ataladi. Notekis massivlar o‘zining ichida boshqa massivlarni element sifatida saqlaydi. Ya’ni, ular birinchi bosqichda bir o‘lchamli massiv sifatida ko‘rinadi, biroq uning har bir elementi o‘z navbatida boshqa bir o‘lchamli massiv bo‘lishi mumkin. Bu massivlarning asosiy xususiyati shundaki, ichki massivlar har xil uzunlikda bo‘lishi mumkin, ya’ni ular “notekis” shaklga ega bo‘ladi.

Notekis massivlar quyidagicha e’lon qilinadi:

int[][] baholar = new int[3][];

```
baholar[0] = new int[2];
baholar[1] = new int[4];
baholar[2] = new int[3];
```

Yuqoridagi misolda baholar nomli massiv 3 ta elementdan iborat bo‘lib, har bir elementi turli uzunlikdagi ichki massivdir. Ya’ni, birinchi satrda 2 ta, ikkinchisida 4 ta, uchinchisida esa 3 ta element mavjud. Bu turdagi massivlar ayniqsa foydalidir, agar har bir guruh yoki to‘plamning elementlar soni oldindan noma’lum yoki har xil bo‘lsa. Misol uchun, turli fanlardan har xil sonli test savollariga ega bo‘lgan talabalar javoblarini saqlashda notekis massivlar ancha mos tushadi.

Notekis massivlar va ko‘p o‘lchovli massivlar o‘rtasidagi asosiy farqlardan biri — **tuzilma soddaligi va moslashuvchanlik darajasi**dir. Ko‘p o‘lchovli massivlarda barcha satrlar va ustunlar bir xil o‘lchamga ega bo‘lishi shart bo‘lsa, notekis massivlarda bu cheklov mavjud emas. Bu esa ularni dinamik ma’lumotlar bilan ishlashda yanada qulay qiladi.

Yana bir muhim farq — **xotira boshqaruvi**da namoyon bo‘ladi. Ko‘p o‘lchovli massivlar xotirada uzluksiz joy egallaydi, bu esa ishlash tezligiga ijobiy ta’sir qilishi mumkin, ayniqsa katta hajmdagi massivlar bilan ishlaganda. Aksincha, notekis massivlarda har bir ichki massiv alohida xotira manzilida saqlanadi. Natijada, ular ko‘proq xotira moslashuvchanligini taklif etadi, biroq ba’zida ishlash tezligi biroz pasayishi mumkin.

Notekis massivlarga murojaat qilishda ikki qadamli indekslashdan foydalaniladi. Masalan, baholar[1][2] ifodasi ikkinchi massivning uchinchi elementini bildiradi. Bunda har ikkala indeks ham 0 dan boshlanadi. Bu indekslash ko‘rinishi dasturchidan e’tibor va aniqlikni talab qiladi, chunki noto‘g‘ri indekslar ishlatilganda *runtime* xatolarga olib kelishi mumkin.

Yakuniy tarzda ta’kidlash joizki, notekis massivlar real hayotdagi murakkab va o‘zgaruvchan tuzilmalarga moslashish imkonini beruvchi kuchli vositadir. Ular C# tilining

obyektga yo‘naltirilgan xususiyatlari bilan uyg‘un ishlaydi va dasturlash amaliyotida keng qo‘llaniladi, ayniqsa foydalanuvchilardan turli uzunlikdagi ma’lumotlarni qayta ishlash talab etilgan hollarda.

Dasturlashda ma’lumotlar bilan samarali ishlash — dastur ishonchliligi va unumdorligining asosiy omillaridan biridir. Shu nuqtai nazardan, massivlar, ayniqsa ko‘p o‘lchovli va notekis massivlar, turli sohalardagi amaliy masalalarni hal etishda keng qo‘llaniladi. Har ikki turdagi massivlar o‘ziga xos afzallik va cheklovlarga ega bo‘lib, ular konkret masalaga qarab tanlanadi.

Ko‘p o‘lchovli massivlar, ayniqsa, **matematik va statistik hisoblashlar**da muhim rol o‘ynaydi. Masalan, ikkita matritsani ko‘paytirish, determinant topish, grafik tizimlarda tasvirni piksel bo‘yicha tahlil qilish, issiqlik xaritalarini modellashtirish kabi jarayonlarda ikki yoki undan ortiq o‘lchovli massivlar ishlatiladi. Chunki bu massivlar tartibli va simmetrik tuzilishga ega, bu esa ularni matematik formulalar bilan uyg‘unlashtirishni osonlashtiradi.

Notekis massivlar esa **foydalanuvchi ma’lumotlari turlicha bo‘lgan holatlar**da afzallik beradi. Masalan, onlayn test tizimlarida har bir talabaga turli sonli savollar to‘g‘ri kelishi mumkin. Yoki biror IT kompaniyada xodimlar turli loyihalarda qatnashadi va har bir xodimda har xil sonli topshiriqlar mavjud bo‘lishi mumkin. Bunday holatlarda bir xil o‘lchamdagi massiv yetarli bo‘lmaydi. Shu sababli, notekis massivlar yordamida **dinamik ma’lumotlar tuzilmalari** tashkil qilinadi.

Ustunliklariga to‘xtalsak, ko‘p o‘lchovli massivlar:

- Tizimli va tartibli ma’lumotlar bilan ishlashda juda qulay;
- Xotirada uzluksiz saqlanadi, bu ishlash tezligini oshiradi;
- Ma’lumotlar orasidagi munosabatlarni (satr-ustun orqali) aniqlashda yengillik yaratadi.

Biroq ularning kamchiligi shundaki:

- Har bir o‘lcham qat’iy belgilanadi, bu esa moslashuvchanlikni kamaytiradi;
- Ortiqcha joy ajratilishi holatlari yuzaga kelishi mumkin.

Notekis massivlar esa:

- Turli uzunlikdagi ma’lumotlar bilan ishlashda o‘zini oqlaydi;
- Har bir ichki massiv mustaqil tuzilishga ega bo‘lib, ehtiyojga ko‘ra shakllantiriladi;
- Dinamik ma’lumotlar strukturasini yaratishga imkon beradi.

Shu bilan birga, notekis massivlarning ham ayrim cheklovlari mavjud:

- Har bir ichki massiv alohida xotira maydonida saqlanadi, bu esa **xotira fragmentatsiyasiga** olib kelishi mumkin;
- Murakkab tuzilma tufayli, ularga ishlov berish, ayniqsa indekslash, murakkablik tug‘diradi;
- Ma’lumotlar ketma-ketligi yo‘qolgan holatlar yuzaga kelishi mumkin.

Amaliyotda ko‘plab dasturchilar massiv tanlashda **moslashuvchanlik va ishlash tezligi o‘rtasidagi muvozanat**ni inobatga oladilar. Shu sababli, dastur tuzilmasi, foydalanuvchi ehtiyoji va ma’lumotlar tabiati asosida massiv turi tanlanadi. Masalan, mobil ilovalar uchun yengil, tezkor ishlaydigan massivlar zarur bo‘lsa, server tomondagi tahliliy hisob-kitoblarda massivlarning quvvati va strukturaviy imkoniyatlari muhim bo‘ladi.

Zamonaviy dasturlashda murakkab ma’lumotlar tuzilmalarini yaratish va boshqarish muhim o‘rin tutadi. Ayniqsa, C# dasturlash tilida mavjud bo‘lgan imkoniyatlar orqali massivlar bilan ishlash ancha qulay va samarali amalga oshiriladi. Shu jihatdan ko‘p o‘lchovli va notekis massivlar dasturchiga har xil murakkablikdagi strukturalarni modellashtirish imkonini beradi.

Ko‘p o‘lchovli massivlar — **aniq struktura va qat’iy o‘lchamlar asosida tuzilgan** massivlar hisoblanadi. Ular, asosan, matematik va vizual modellashtirishlarda, statistik tahlil, raqamli signallarni qayta ishlash, fizik jarayonlarni kompyuter modellashtirish kabi

sohalarda keng qo‘llaniladi. Bu massivlar tizimli va izchil tuzilishga ega bo‘lgani sababli, algoritmik jihatdan ularni qayta ishlash nisbatan oson va aniqdir.

Boshqa tomondan, notekis massivlar esa dasturiy yechimlarda **moslashuvchanlik va sharoitga qarab shakllanish** imkonini taqdim etadi. Turli foydalanuvchi guruhlari uchun moslashgan ma’lumotlar, nomutanosib kiruvchi ma’lumotlar oqimi yoki har xil uzunlikdagi ro‘yxatlar bilan ishlaganda notekis massivlar eng maqbul tanlov bo‘lib xizmat qiladi. Aynan shuning uchun ham ularning qo‘llanish doirasi keng bo‘lib, veb-ilovalar, foydalanuvchi interfeyslari, test tizimlari va ma’lumotlar bazasi bilan integratsiyalashgan modullarni yaratishda keng ishlatiladi.

Texnik nuqtai nazardan, har ikkala massiv turining o‘ziga xos tuzilmasi, xotira tashkiloti va indekslash mexanizmlari mavjud. Ko‘p o‘lchovli massivlar xotirada ketma-ket, bir butun blokda saqlansa, notekis massivlar har bir ichki elementni alohida massiv sifatida ko‘radi va saqlaydi. Bu jihat ularning ishlash samaradorligiga bevosita ta’sir ko‘rsatadi. Shu bois, dasturchi har bir konkret holatda **ishlash tezligi, xotira sarfi va tuzilma soddaligi** mezonlarini hisobga olgan holda tanlov qilishi zarur.

Shuningdek, massivlar bilan ishlashda **indeksdan foydalanishning to‘g‘riligini tekshirish, xatoliklar bilan ishlash, dinamik ajratish va ma’lumotlarga murojaat tezligi** kabi texnik aspektlar e’tibordan chetda qolmasligi lozim. Ayniqsa notekis massivlarda noto‘g‘ri indekslash Index Out Of Range Exception xatosiga olib kelishi mumkin.

Yakunda aytish mumkinki, C# dasturlash tilida ko‘p o‘lchovli va notekis massivlar orqali murakkab ma’lumotlar tuzilmalari samarali ifodalanadi. Ularning o‘zaro farqlari, imkoniyatlari va cheklovlarini chuqur o‘rganish — dasturchining muammoga to‘g‘ri yondashuvi va optimal yechim tanlashi uchun zamin yaratadi. Shuningdek, bu bilimlar dasturchilarni nafaqat texnik saviyaga olib chiqadi, balki ularni amaliyotga yo‘naltirilgan tahliliy fikrlashga ham undaydi.

Xulosa

C# dasturlash tilida ko‘p o‘lchovli va notekis massivlar dasturchilar uchun kuchli va moslashuvchan ma’lumotlar tuzilmasi sifatida xizmat qiladi. Ushbu maqolada har ikki massiv turi to‘g‘risidagi nazariy tushunchalar, ularning tuzilish mexanizmi, dasturiy amaliyotdagi o‘rni hamda o‘zaro farqlari batafsil tahlil qilindi. Ko‘p o‘lchovli massivlar ma’lumotlarni qat’iy o‘lcham va tuzilishga ega bo‘lgan holatlarda saqlash uchun mo‘ljallangan bo‘lib, ayniqsa matritsalar, koordinatalar tizimi, grafik tasvirlar va matematik modellar bilan ishlashda alohida o‘rin tutadi.

Notekis massivlar esa, o‘z tabiatiga ko‘ra, ichki tuzilmalarning har xil uzunlikda bo‘lishiga imkon beruvchi moslashuvchan struktura hisoblanadi. Bu massivlar foydalanuvchi kiritgan ma’lumotlar hajmi va turi oldindan noma’lum yoki o‘zgaruvchan bo‘lgan holatlarda eng maqbul yechim sanaladi. Xususan, test tizimlari, javoblar ro‘yxati, dinamik jadval strukturalari va ko‘p darajali konfiguratsiyalarga ega bo‘lgan ilovalarda notekis massivlardan foydalanish samarali natijalar beradi.

Shuningdek, maqolada massivlarning texnik jihatlari, jumladan xotirada joylashuvi, ishlash tezligi, indekslash mexanizmi, xatoliklar bilan ishlash muammolari ham yoritildi. Dasturchi uchun har ikki massiv turining imkoniyatlarini chuqur tahlil qilish, konkret vazifaga eng mos tuzilmani tanlash, dastur samaradorligi va ishonchliligini oshirishda muhim ahamiyat kasb etadi. C# dasturlash tilining kuchli obyektga yo‘naltirilgan xususiyatlari massivlar bilan ishlashni yanada qulaylashtiradi hamda murakkab strukturaviy ma’lumotlarni modellashtirish imkonini yaratadi.

Ko‘p o‘lchovli va notekis massivlar C# dasturlash muhitining asosiy vositalaridan biri bo‘lib, ular bilan samarali ishlash dasturchining nazariy bilimlari va amaliy ko‘nikmalariga bevosita bog‘liq. Ushbu bilimlarni puxta egallash esa zamonaviy dasturiy mahsulotlar yaratishda muvaffaqiyat kalitidir.

Adabiyotlar ro‘yxati:

1. Шарипов, Ш. Ш. (2020). *Algoritmlar va dasturlash asoslari*. Toshkent: Fan va texnologiya nashriyoti.
2. Насруллаев, Р. (2018). *C# dasturlash tili asoslari*. Toshkent: “Iqtisodiyot” nashriyoti.
3. Бобоев, Б. (2021). *Massivlar va ular bilan ishlashning amaliy jihatlari*. TATU Ilmiy jurnali, 2(1), 34–41.
4. Алимов, А. Р., Абдурахмонов, А. И. (2017). *Informatika va axborot texnologiyalari*. Toshkent: Oliy ta’lim nashriyoti.
5. Albahari, J., & Albahari, B. (2021). *C# 10.0 in a Nutshell: The Definitive Reference*. O’Reilly Media.
6. Troelsen, A., & Japikse, P. (2022). *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming*. Apress.
7. Richter, J. (2012). *CLR via C#*. Microsoft Press.
8. Skeet, J. (2019). *C# in Depth*. Manning Publications.
9. Lippert, E. (2011). *C# Programming Guide* — Microsoft Docs. https://learn.microsoft.com/en-us/dotnet/csharp/