NAVBAT VA STEK BILAN ISHLASH

Yusupov Mirsaid Abdulaziz o'g'li
Farg'ona davlat universiteti kafedra o'qituvchisi
mirsaidbeky@gmail.com
A'zamova Sevinch
Farg'ona davlat universiteti 2-kurs talabasi
dilafruzahmadjonova4@gmail.com

Annotatsiya: Mazkur maqolada zamonaviy kompyuter fanining muhim bo'g'inlaridan biri bo'lgan ma'lumotlar tuzilmalari, xususan, stek va navbatning nazariy va amaliy asoslari tahlil qilinadi. Ushbu tuzilmalar algoritmik muammolarni hal qilishda qanday ishlatilishi, ularning LIFO va FIFO prinsiplariga asoslangan xususiyatlari, shuningdek, C# dasturlash tilida ular bilan qanday ishlash mumkinligi batafsil bayon etilgan. Real hayotdagi dasturiy tizimlarda stek va navbatning roli, ularning afzalliklari va cheklovlari misollar orqali yoritiladi. Dasturiy kod namunasi orqali o'quvchiga tuzilmalarning ishlash mexanizmi amaliy tarzda tushuntiriladi. Ushbu maqola informatikani o'rganayotgan talabalar, dasturchilar va algoritmlar bilan ishlovchi tadqiqotchilar uchun foydalidir.

Kalit so'zlar: stek, navbat, ma'lumotlar tuzilmasi, algoritm, C# dasturlash tili, generik klasslar, LIFO, FIFO, dasturiy modellashtirish, funksiyalar steki, navbatga qo'yish, operatsion tizimlar, asinxron tizimlar, rekursiya, sintaksis tahlil, kompyuter fanlari

Аннотация: В статье рассматриваются теоретические и практические аспекты структур данных, таких как стек и очередь, которые являются неотъемлемой частью современной информатики. Подробно анализируются принципы их работы — LIFO и FIFO, а также способы реализации на языке программирования C#. Представлены примеры использования этих структур в реальных программных системах и рассмотрены их преимущества и ограничения. Иллюстрация с примером кода помогает читателю глубже понять механизмы работы структур. Материал будет полезен студентам, изучающим информатику, начинающим разработчикам и исследователям, работающим с алгоритмами.

Ключевые слова: стек, очередь, структура данных, алгоритм, язык программирования C#, обобщённые классы, LIFO, FIFO, программное моделирование, стек вызовов, диспетчеризация процессов, асинхронные системы, рекурсия, синтаксический анализ, прикладная информатика

Annotation: This article explores the theoretical and practical foundations of key data structures — namely, stack and queue — which play a pivotal role in modern computer science. It discusses their operational principles based on LIFO and FIFO

mechanisms, implementation strategies in the C# programming language, and their application in real-world software systems. The article highlights both the advantages and limitations of these structures through analytical explanation and a practical code example. The content is particularly useful for computer science students, software developers, and researchers working in the field of algorithms and data structures.

Keywords: stack, queue, data structure, algorithm, C# programming language, generic classes, LIFO, FIFO, software modeling, call stack, process scheduling, asynchronous systems, recursion, syntax parsing, applied computer science

Kirish

Axborotni qayta ishlash, saqlash va uzatish jarayonlarida ma’lumotlarning vaqtinchalik tartibli boshqaruvi muhim ahamiyat kasb etadi. Kompyuter fanida bu vazifani bajarishda asosiy ma’lumot tuzilmalaridan biri sifatida **navbat (queue)** va **stek (stack)** strukturalari keng qo‘llaniladi. Ushbu tuzilmalar algoritmlarni optimallashtirish, operativ xotira bilan ishlash va dasturiy jarayonlarni samarali boshqarishda muhim vositalardir.

Stek va navbat o‘zaro farqli tartibda ma’lumotlarni saqlaydi: stek tamoyili "oxirgisi birinchi" (LIFO — Last In, First Out) asosida ishlasa, navbat "birinchisi birinchi" (FIFO — First In, First Out) prinsipiga asoslanadi. Bu ikki tuzilma yordamida dasturlashda funksiyalarni chaqirish, resurslar ustida navbat tartibida ishlash, oraliq hisob-kitoblarni saqlab qolish kabi muhim amallar bajariladi.

Mazkur maqolada navbat va stek ma’lumot tuzilmalarining nazariy asoslari, ularning ishlash mexanizmlari, amaliy qo‘llanilishi va dasturlashdagi o‘rni batafsil tahlil qilinadi. Shuningdek, ular bilan ishlashning samarali algoritmlari, dasturiy realizatsiya usullari va real hayotdagi qo‘llanilish misollari ko‘rib chiqiladi.

Navbat va stek tuzilmalarining nazariy asosi

Zamonaviy informatikaning muhim yo‘nalishlaridan biri — ma’lumotlar tuzilmalarini samarali tashkil qilishdir. Bunda **chiziqli tuzilmalar** — xususan, **stek (stack)** va **navbat (queue)** alohida ahamiyat kasb etadi. Ular kompyuter tizimlarining ham dasturiy, ham apparat darajasidagi samaradorligini belgilovchi poydevor elementlaridan biridir

Dasturlashda ma’lumotlarni vaqtinchalik saqlash va ularga navbat bilan yoki teskari tartibda murojaat qilish zarurati tug‘iladi. Shu maqsadda **Stack** (stek) va **Queue** (navbat) kabi **chiziqli ma’lumotlar tuzilmalaridan** foydalaniladi. Ular C# dasturlash tilida System.Collections va System.Collections.Generic nomli kutubxonalar orqali qo‘llab-quvvatlanadi va ma’lumotlarni o‘ziga xos tartibda boshqarishga yordam beradi.

Stek — bu ma’lumotlar bilan "oxirgisi birinchi chiqadi") prinsipida ishlovchi chiziqli tuzilma bo‘lib, unda ma’lumotlar ketma-ketlikda qo‘shiladi va chiqariladi. Bu struktura dasturlashda funksiya chaqiruvlari stekini saqlashda, teskari tartibdagi

amallarni bajarishda, masalan, ifodalarni postfix formatga o‘tkazishda keng qo‘llaniladi

Unda quyidagi asosiy metodlar mavjud:

- Push(item) – stekning yuqori qismiga element qo‘shadi.
- Pop() – stekning yuqorisidagi elementni olib tashlaydi va qaytaradi.
- Peek() – stekning yuqorisidagi elementni ko‘rsatadi, lekin olib tashlamaydi.

Bu tuzilma dasturlashda **funksiya chaqiruvlarini kuzatish**, **teskari tartibda chiqishlar**, yoki **qavslar muvofiqligini tekshirish** kabi holatlarda ishlatiladi.

Navbat esa "birinchisi birinchi chiqadi" prinsipiga asoslangan bo‘lib, bunda birinchi qo‘shilgan element birinchi bo‘lib chiqariladi. Navbatlar tarmoq paketlarini uzatishda, jarayonlarni boshqarishda, foydalanuvchi so‘rovlarini navbatga qo‘yishda va real vaqtda ishlov berish tizimlarida muhim rol o‘ynaydi

Undagi asosiy metodlar:

- Enqueue(item) – navbat oxiriga element qo‘shadi.
- Dequeue() – navbat boshidagi elementni olib tashlaydi va qaytaradi.
- Peek() – navbat boshidagi elementni ko‘rsatadi, lekin olib tashlamaydi.

Navbatdan odatda **printer navbatlarini boshqarish**, **tartib bilan bajariladigan ishlar** yoki **jarayonlarni ketma-ket rejalashtirish**da foydalaniladi.

Stek va navbatning C# tilidagi amaliy ifodasi

C# dasturlash tilida System.Collections.Generic nomli kutubxona orqali Stack<T> va Queue<T> generik sinflari orqali ushbu tuzilmalar bilan samarali ishlash mumkin. Quyida **stek** bilan ishlashga oid bitta namuna keltiriladi:

```
using System;
using System.Collections.Generic;

class StekMisol
{
   static void Main()
   {
      Stack<string> stek = new Stack<string>();

      stek.Push("Algoritm");
      stek.Push("Ma'lumotlar tuzilmasi");
      stek.Push("Stek");

      Console.WriteLine("Yuqoridagi element: " + stek.Peek());

      while (stek.Count > 0)
      {
         Console.WriteLine("Chiqarildi: " + stek.Pop());
```

```
        }
    }
}
```

Mazkur kod namunasi Push(), Peek(), va Pop() metodlarining funksionalligini yaqqol ifodalaydi. Push() orqali yangi element stekga joylashtiriladi, Peek() orqali eng yuqoridagi element ko‘riladi (ammo olib tashlanmaydi), Pop() esa elementni olib, stekdan chiqaradi.

Quyidagi misolda bitta konsol dasturi orqali Stack va Queue bilan ishlash ko‘rsatilgan:

```
using System;
using System.Collections.Generic;

namespace StackQueueExample
{
    class Program
    {
        static void Main(string[] args)
        {
            // Stek bilan ishlash
            Stack<string> stek = new Stack<string>();
            stek.Push("Birinchi");
            stek.Push("Ikkinchi");
            stek.Push("Uchinchi");

            Console.WriteLine("Stekdagi elementlar:");
            while (stek.Count > 0)
            {
                Console.WriteLine(stek.Pop());
            }

            Queue<string> navbat = new Queue<string>();
            navbat.Enqueue("Ali");
            navbat.Enqueue("Vali");
            navbat.Enqueue("Karim");

            Console.WriteLine("\nNavbatdagi elementlar:");
            while (navbat.Count > 0)
            {
                Console.WriteLine(navbat.Dequeue());
```

```
            }
        }
    }
}
```

Yuqoridagi dastur Stack va Queue tuzilmalarining amaliy qo‘llanilishini ochiq ko‘rsatadi. Stekda "Uchinchi" elementi birinchi chiqariladi, chunki u oxirgi qo‘shilgan. Aksincha, navbatda "Ali" birinchi chiqariladi, chunki u birinchi bo‘lib kiritilgan. Bu printsiplar ma’lumotlar oqimini boshqarishda va vaqtinchalik saqlashda juda muhimdir.

Stack (Steck) – Oxirgi kirgan birinchi chiqadi

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
        Stack<string> kitoblar = new Stack<string>();
        kitoblar.Push("Matematika");
        kitoblar.Push("Fizika");
        kitoblar.Push("Ingliz tili");

        Console.WriteLine("Oxirgi kitob: " + kitoblar.Pop()); // Ingliz tili
        Console.WriteLine("Keyingi kitob: " + kitoblar.Pop()); // Fizika
    }
}
```

Kitoblar ustma-ust qo‘yilgan. Oxirgi qo‘yilgan (Ingliz tili) birinchi olinadi.

Queue (Navbat) – Birinchi kirgan birinchi chiqadi

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
        Queue<string> navbat = new Queue<string>();
        navbat.Enqueue("Ali");
        navbat.Enqueue("Vali");
        navbat.Enqueue("Karim");
```

```
        Console.WriteLine("Navbatdagisi: " + navbat.Dequeue()); // Ali
        Console.WriteLine("Keyingi: " + navbat.Dequeue()); // Vali
    }
}
```

Bu yerda odamlar navbatga turishgan. Birinchi kelgan birinchi xizmat oladi (Ali → Vali → Karim).Amaliy jihatlar va dasturiy muhitda qo‘llanish

Stek tuzilmasi quyidagi sohalarda o‘zining amaliy ahamiyatini isbotlagan:

- **Rekursiv funksiyalarni boshqarish**: har bir funksiyaning lokal konteksti stek orqali saqlanadi;
- **Sintaksis tahlil (parsing)**: arifmetik yoki mantiqiy ifodalarni tekshirishda (masalan, qavslar muvozanatini aniqlash);
- **Postfix va prefix ifodalar**: odatiy (infix) ifodalarni teskari yoki oldin yozilgan formatlarga o‘tkazishda;

Navbat tuzilmasi esa quyidagi real tizimlarda keng qo‘llaniladi:

- **Protsessorlar uchun navbat rejimi (CPU scheduling)**;
- **Ma’lumotlar oqimini boshqarish (Data streaming)**;
- **Asinxron tizimlar va hodisalar navbati (event-driven programming)**;
- **Shahar transport tizimlarida navbatda boshqaruv algoritmlari**.

Bu tuzilmalar yuqorida keltirilgan hollarda, samarali algoritmlarni yaratish va tizim ish faoliyatini optimallashtirishda asosiy vosita bo‘lib xizmat qiladi.

Navbat va stek bir qarashda oddiy ko‘rinadigan strukturalar bo‘lishiga qaramasdan, ular algoritmlarni modellashtirishda fundamental ahamiyatga ega. Aynan ushbu tuzilmalar orqali murakkab dasturiy oqimlar, real vaqtda ishlov beruvchi tizimlar va operatsion tizimlar arxitekturasi soddalashtiriladi. Donald Knut o‘zining mashhur "The Art of Computer Programming" asarida stekni "algoritmik fikrlashda zaruriy vosita" deb baholaydi (Knuth, 1997).

Stek — vaqtinchalik saqlash, navbat esa ketma-ketlikda xizmat ko‘rsatish uchun eng optimal modeldir. Ularni noto‘g‘ri tanlash algoritmni haddan tashqari murakkablashtiradi yoki resurslarni samarasiz boshqarilishiga olib keladi. Shu bois, ularni to‘g‘ri qo‘llash har qanday dasturiy tizimning muhim jihatlaridan biridir.

Stek ma’lumotlar tuzilmasi dasturiy ta’minotda keng ko‘lamda qo‘llaniladi. Misol uchun, **funksiyalar chaqiruv steki (call stack)** C# kabi ko‘plab tillarda kod ishlayotganida funksiyalarni kuzatib boradi. Har bir yangi funksiya chaqiruvi stekga qo‘shiladi, bajarilgach esa chiqariladi. Shu tarzda dastur rekursiv algoritmlarni boshqaradi. Masalan, **rekursiv Fibbonachi algoritmlari, DFS (depth-first search)** kabi algoritmlar stekdan faol foydalanadi.

Yana bir misol — **veb brauzerda sahifalarni orqaga va oldinga yurish** funksiyasi. Har safar foydalanuvchi yangi sahifaga o‘tganda, joriy sahifa stekga qo‘shiladi. Orqaga tugmasi bosilganda, stekdan element chiqariladi va oldingi sahifa yuklanadi.

Navbat esa ko‘proq **tizim resurslarini boshqarish**da ishlatiladi. Misol uchun:

- **Printerlar navbati**: har bir chiqish vazifasi tartib bilan navbatga qo‘yiladi va ketma-ket bajariladi.
- **Protsessor jarayonlarini boshqarish (CPU scheduling)**: operatsion tizim turli jarayonlarni navbatga joylashtirib, ularga navbat bilan ishlov beradi.
- **Mijozlarga xizmat ko‘rsatish tizimlari (call center, bank xizmatlari)** ham navbat asosida tashkil etiladi.

Shuningdek, **asinxron tizimlarda navbat** vositasi ko‘p uchraydi. Masalan, biror veb ilovada foydalanuvchi so‘rov yuboradi, u navbatga tushadi va backend tizimi ularni ketma-ket bajaradi

Stek va navbat algoritmlarining samarali ishlashi

Stek va navbat bilan ishlashda **algoritmik samaradorlik** juda muhim. Har ikki tuzilmada asosiy operatsiyalar — Push, Pop, Enqueue, Dequeue — **O(1)** vaqt murakkabligida bajariladi. Bu ularni juda samarali qiladi.

Misol:

- **Balanslangan qavslar tekshiruvi**: har bir ochuvchi qavs Push qilinadi, yopuvchi qavs kelganda Pop qilinib, mosligi tekshiriladi.
- **Palindrom aniqlash**: so‘zlarni stekga qo‘yib, teskari tartibda solishtirish mumkin.
- **BFS algoritmi (navbat asosida)** orqali grafni bosqichma-bosqich aylanib chiqish mumkin.

Stek va navbat ma’lumotlar tuzilmalari zamonaviy dasturlashning ajralmas qismidir. Ular orqali nafaqat kodni to‘g‘ri strukturalash, balki tizim resurslarini samarali boshqarish ham mumkin. Dasturchilar ushbu tuzilmalarni chuqur tushunsa, murakkab tizimlar yaratishda asosiy tayanchga ega bo‘ladi. Ayniqsa, algoritmik musobaqalar, amaliy loyihalar va real tizimlar bilan ishlashda bu bilimlar alohida ustunlik keltiradi.

Navbat va stek — dasturlashda tez-tez uchraydigan, lekin mohiyatan chuqur algoritmik strukturadir. Ularning samarali qo‘llanilishi dasturiy ta’minotning funksional, barqaror va optimalligi uchun muhim ahamiyatga ega. Stek vaqtinchalik saqlash va teskari tartibda ishlov berish jarayonlarida, navbat esa tartibga solingan ketma-ket xizmat ko‘rsatish tizimlarida keng qo‘llaniladi. C# dasturlash tilida ushbu strukturalarning generik klasslar orqali soddalashtirilgan ifodasi mavjud bo‘lib, ular bilan ishlash oson va qulay. Maqolada keltirilgan nazariy va amaliy tahlillar ushbu strukturalarning real tizimlarda qo‘llanilishining asosli va zarurligini ko‘rsatadi.

Foydalanilgan adabiyotlar:

1. Knuth D. E. *The Art of Computer Programming. Volume 1: Fundamental Algorithms*. – Addison-Wesley, 1997.

2. Aho A. V., Hopcroft J. E., Ullman J. D. *Data Structures and Algorithms*. – Addison-Wesley, 1983.
3. Tanenbaum A. S., Bos H. *Modern Operating Systems*. – Pearson Education, 2015.
4. Lafore R. *Data Structures and Algorithms in C++*. – Sams Publishing, 2002.
5. Sedgewick R., Wayne K. *Algorithms (4th Edition)*. – Addison-Wesley, 2011.
6. Richter J. *CLR via C# (4th Edition)*. – Microsoft Press, 2012.
7. Миловидов К. А., Панюков В. А. *Алгоритмы и структуры данных*. – М.: БХВ-Петербург, 2019.
8. Karimov I. S., Jo‘raqulov M. *Dasturlash asoslari (C# tili misolida)*. – Toshkent: Innovatsion rivojlanish nashriyoti, 2020.
9. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/ – Rasmiy C# hujjatlari va kod namunalar bazasi.
10. **Weiss M. A.** *Data Structures and Algorithm Analysis in C++*. – Pearson, 2014.
11. **Troelsen A., Japikse P.** *Pro C# 10 with .NET 6.* – Apress, 2022.
12. **Knott G. D.** *Data Structures and Algorithms Using C#.* – Addison Wesley, 2004.
13. **Basharov M. Sh.** *Algoritmlar va ularning C# tilidagi dasturiy realizatsiyasi.* – Toshkent: Fan va texnologiyalar, 2021.