***Mavzu: Navbat. Navbatni massiv yordamida tasvirlash va ular ustida amal bajarish algoritmlari.***

Navbat (yoki "navbatlar") - ma'lumotlarni saqlash va boshqarish uchun ishlatiladigan strukturalardan biri. Navbatlar ko'pincha FIFO (First In, First Out) prinsipiga asoslanadi, ya'ni birinchi bo'lib kirgan element birinchi bo'lib chiqariladi. Navbatni massiv yordamida tasvirlash va ularda amal bajarish algoritmlari haqida quyida ma'lumot beraman.

**Navbat tushunchasi:**

1. **Tashkiliy tuzilma**: Navbatlar odamlar yoki jarayonlar o'rtasida tartibni saqlash uchun tashkil etiladi.
2. **Resurslarni taqsimlash**: Navbat yordamida cheklangan resurslar samarali taqsimlanadi, bu esa xizmat ko'rsatish sifatini oshiradi.

**Muhimligi:**

1. **Samaradorlikni oshirish**: Navbatlar muayyan tartibda xizmat ko'rsatish imkonini beradi, bu esa umumiy samaradorlikni oshiradi.
2. **Xizmat sifatini yaxshilash**: Mijozlar o'z navbatini kutayotganida, xizmat ko'rsatish jarayoni yanada sifatli bo'ladi.
3. **O'zaro munosabatlarni boshqarish**: Navbatlar odamlar o'rtasidagi munosabatlarni tartibga solishga yordam beradi, bu esa ijtimoiy tartibni saqlashga olib keladi.
4. **Iqtisodiy jihatdan samaradorlik**: Navbatli xizmat ko'rsatish tizimlari iqtisodiy jihatdan samarali bo'lib, resurslarni optimallashtirishga yordam beradi.

Umuman olganda, navbat tushunchasi har qanday jamiyatda muhim rol o'ynaydi va uni samarali boshqarish hayot sifatini oshirishga olib keladi.

**Navbatni Massiv Yordamida Tasvirlash**

Massiv yordamida navbatni tasvirlash uchun, biz bir nechta o'zgaruvchilarni kiritamiz:

1. **massiv**: Navbatni saqlash uchun.
2. **front**: Navbatning boshini ko'rsatadigan indeks.
3. **rear**: Navbatning oxirini ko'rsatadigan indeks.
4. **maxSize**: Navbatning maksimal o'lchami.
5. **currentSize**: Hozirgi elementlar soni.

### Massiv bilan Navbatni Boshqarish Algoritmlari

#### 1. Navbatni yaratish

class Queue:

def \_\_init\_\_(self, maxSize):

self.maxSize = maxSize

self.queue = [None] \* maxSize

self.front = 0

self.rear = -1

self.currentSize = 0

2. Navbatga qo'shish (enqueue)

def enqueue(self, item):

if self.currentSize == self.maxSize:

print("Navbat to'lib ketdi!")

return

self.rear = (self.rear + 1) % self.maxSize

self.queue[self.rear] = item

self.currentSize += 1

3. Navbatdan chiqarish (dequeue)

def dequeue(self):

if self.currentSize == 0:

print("Navbat bo'sh!")

return None

item = self.queue[self.front]

self.front = (self.front + 1) % self.maxSize

self.currentSize -= 1

return item

4. Navbatning boshidagi elementni ko'rish (peek)

def peek(self):

if self.currentSize == 0:

print("Navbat bo'sh!")

return None

return self.queue[self.front]

1. Navbatni tekshirish (is\_empty va is\_full)

Navbatning bo'sh yoki to'la ekanligini tekshirish uchun quyidagi funksiyalarni yozish mumkin:

***def is\_empty():***

***return size == 0***

***def is\_full():***

***return size == max\_size***

***Navbatga Element Qo'shish (Enqueue)***

***Enqueue amali navbatga yangi element qo'shadi. Bu amal oxirgi ko'rsatkichni ishlatadi.***

***Bo'sh joy tekshirish***

***Navbatda bo'sh joy borligini tekshirish kerak. Agar to'la bo'lsa, xato qaytariladi.***

***Element qo'shish***

***Yangi element oxirgi ko'rsatkich pozitsiyasiga qo'yiladi. Keyin ko'rsatkich oshiriladi.***

***Navbat holatini yangilash***

***Oxirgi ko'rsatkich yangilanadi. Agar birinchi element bo'lsa, bosh ko'rsatkich ham o'rnatiladi.***

***Navbatdan Element Olish (Dequeue)***

***Dequeue amali navbatdan birinchi elementni olib tashlaydi. Bu amal bosh ko'rsatkichni ishlatadi.***

***Bo'sh emaslikni tekshirish***

***Navbat bo'sh emasligini tekshirish kerak. Agar bo'sh bo'lsa, xato qaytariladi.***

***Elementni olish***

***Bosh ko'rsatkich pozitsiyasidagi element olinadi. So'ng ko'rsatkich keyingi elementga siljitiladi.***

***Navbat holatini yangilash***

***Bosh ko'rsatkich yangilanadi. Agar oxirgi element bo'lsa, navbat bo'sh holatga o'tkaziladi.***

***Navbatni Tekshirish Amallari***

***Bo'shlikni tekshirish***

***Bosh va oxirgi ko'rsatkichlar teng bo'lsa, navbat bo'sh hisoblanadi.***

***To'liqlikni tekshirish***

***Oxirgi ko'rsatkich massiv oxiriga yetsa, navbat to'la hisoblanadi.***

***Elementlar sonini hisoblash***

***Bosh va oxirgi ko'rsatkichlar orasidagi farq elementlar sonini beradi.***

***Navbatni Tasvirlash***

***Navbatni tasvirlash uchun bosh ko'rsatkichdan boshlab oxirgi ko'rsatkichgacha elementlar chop etiladi. Bu navbat holatini ko'rsatadi.***

***Navbat Amallari Murakkabligi***

***Amal***

***Vaqt murakkabligi***

***Enqueue***

***O(1)***

***Dequeue***

***O(1)***

***Bo'shlikni tekshirish***

***O(1)***

***To'liqlikni tekshirish***

***O(1)***

***Navbat Qo'llanilishi***

***Navbatlar turli dasturlash masalalarida qo'llaniladi. Ular jarayonlarni boshqarish, tarmoq buferlari va keng qidiruv algoritmlarida ishlatiladi.***

***Jarayonlar boshqaruvi***

***Operatsion tizimlarda vazifalarni navbatga qo'yish uchun ishlatiladi.***

***Tarmoq buferlari***

***Ma'lumotlarni vaqtincha saqlash va uzatish uchun foydalaniladi.***

***Qidiruv algoritmlari***

***Graflar ustida keng qidiruv (BFS) algoritmida qo'llaniladi.***

Navbatning ishlash prinsipi FIFO (First In, First Out) — "birinchi kirmoq, birinchi chiqmoq" tamoyiliga asoslangan. Bu prinsip asosan ma'lumotlar tuzilmalarida, hisoblash tizimlarida va xizmat ko'rsatishda qo'llaniladi. FIFO tamoyili quyidagi xususiyatlarga ega:

**Asosiy xususiyatlari:**

1. **Tartib**: Navbatga kirgan birinchi element birinchi bo'lib chiqariladi. Bu shuni anglatadiki, xizmat ko'rsatish yoki resurslardan foydalanish tartibi saqlanadi.
2. **Adolat**: Har bir kishi yoki ob'ekt o'z navbatini kutish orqali adolatli ravishda xizmat oladi, bu esa xizmat sifatini oshiradi.
3. **Sodda tuzilma**: FIFO tizimi oddiy tuzilishga ega bo'lib, uni amalga oshirish oson. Ma'lumotlar qatorida biriktirilgan elementlar birinchi kirgan elementlardan boshlanadi.

**Qo'llanishi:**

1. **Xizmat ko'rsatish**: Restoranlar, banklar yoki boshqa xizmat ko'rsatish joylarida mijozlar navbatda kutib, o'z navbatlarini olishadi.
2. **Ma'lumotlar bazasi**: Kompyuter tizimlarida FIFO navbatlar ko'pincha ma'lumotlar oqimini boshqarishda ishlatiladi, masalan, printerlar yoki disklar orqali ma'lumotlarni qayta ishlashda.
3. **Ishlab chiqarish**: FIFO ishlab chiqarish jarayonlarida, materiallar va mahsulotlar inventarizatsiyasini boshqarish uchun qo'llaniladi.

**Afzalliklari:**

* **Sodda boshqaruv**: FIFO tizimi oddiy va intuitivdir, bu esa uning boshqaruvini osonlashtiradi.
* **Samaradorlik**: Ushbu tamoyil resurslarni samarali taqsimlash va xizmat ko'rsatish jarayonlarini optimallashtirishga yordam beradi.

FIFO prinsipining samarali ishlashi xizmat ko'rsatish sifatini va tartibini saqlashda muhim ahamiyatga ega.

Massivni e'lon qilish va kerakli o'zgaruvchilarni aniqlash uchun dasturlash tiliga qarab quyidagi misollarni keltirish mumkin. Keling, C yoki C++ dasturlash tillarida FIFO navbatini amalga oshirishda kerakli elementlarni ko'rib chiqamiz.

Massivning o'lchamini belgilash dasturlash tiliga bog'liq bo'lsa-da, ko'pincha massivni e'lon qilishda uning maksimal o'lchamini belgilash kerak. Misol uchun, C yoki C++ dasturlash tillarida quyidagi ko'rinishda amalga oshiriladi.

### Massivning o'lchamini belgilash

C yoki C++ da massivni e'lon qilish uchun #define yoki const kalit so'zlaridan foydalanishingiz mumkin. Mana bir necha misol:

#### Misol 1: #define yordamida

***#define MAX\_SIZE 100 // Navbatning maksimal o'lchami***

***int myArray[MAX\_SIZE]; // Massivni e'lon qilish***

Misol 2: const yordamida

***const int MAX\_SIZE = 100; // Navbatning maksimal o'lchami***

* ***int myArray[MAX\_SIZE]; // Massivni e'lon qilish*** MAX\_SIZE o'zgaruvchisi orqali massivning maksimal o'lchamini belgilash, dasturda moslashuvchanlikni ta'minlaydi.
* Agar siz massivning o'lchamini dastur ishga tushish vaqtida belgilashni xohlasangiz, dinamik xotira ajratish usullaridan (masalan, malloc yoki new) foydalanishingiz mumkin. Bunda massivning o'lchami dastur vaqtida aniqlanadi.

To'g'ri, ko'rsatkich (index) dastlabki qiymati 0 ga o'rnatiladi. Bu ko'rsatkich massivdagi oxirgi elementning joylashgan pozitsiyasini ko'rsatadi. Yangi element qo'shilganda, ko'rsatkich har safar 1 ga oshadi.

Masalan, quyidagi qadamlar orqali ko'rsatkichni boshqarish jarayonini yanada chuqurroq ko'rib chiqamiz:

1. **Boshlang'ich qiymat**: Ko'rsatkichni 0 ga o'rnatamiz.
2. **Element qo'shish**: Yangi element qo'shilganda, ko'rsatkichni massivning uzunligi orqali yangilaymiz.
3. **Oxirgi elementga murojaat**: Har safar yangi element qo'shilganda, massivning oxirgi pozitsiyasiga qo'shamiz.

Quyidagi kodda ko'rsatkichni qanday ishlatishni ko'rsataman:

***# Bo'sh massiv***

***massiv = []***

***# Ko'rsatkich***

***ko'rsatkich = 0***

***# Yangi element qo'shish funksiyasi***

***def yangi\_element\_qosh(massiv, element):***

***global ko'rsatkich***

***massiv.append(element) # Elementni massivga qo'shamiz***

***ko'rsatkich += 1 # Ko'rsatkichni oshiramiz***

***# Misol uchun, yangi elementlarni qo'shish***

***yangi\_element\_qosh(massiv, 10)***

***yangi\_element\_qosh(massiv, 20)***

***print(massiv) # [10, 20]***

***print(ko'rsatkich) # 2***

***# Bo'sh massiv***

***massiv = []***

***# Ko'rsatkich***

***ko'rsatkich = 0***

***# Yangi element qo'shish funksiyasi***

***def yangi\_element\_qosh(massiv, element):***

***global ko'rsatkich***

***massiv.append(element) # Elementni massivga qo'shamiz***

***ko'rsatkich += 1 # Ko'rsatkichni oshiramiz***

***# Misol uchun, yangi elementlarni qo'shish***

***yangi\_element\_qosh(massiv, 10)***

***yangi\_element\_qosh(massiv, 20)***

***print(massiv) # [10, 20]***

***print(ko'rsatkich) # 2***

To‘g‘ri, yangi element qo‘shilganda ko‘rsatkichni oshirish va yangi qiymatni massivga joylash jarayoni juda muhim. Har safar yangi element qo‘shilganda, ko‘rsatkich massivning oxiriga ko‘rsatiladi va keyin yangilanishi kerak.

Mana qanday amalga oshirish mumkin:

1. **Element qo‘shish**: Har safar yangi elementni massivga joylaymiz.
2. **Ko‘rsatkichni oshirish**: Yangi element qo‘shilgandan so‘ng, ko‘rsatkichni 1 ga oshiramiz.

Mana bir misol:

***# Bo'sh massiv***

***massiv = []***

***# Ko'rsatkich***

***ko'rsatkich = 0***

***# Yangi element qo'shish funksiyasi***

***def yangi\_element\_qosh(massiv, element):***

***global ko'rsatkich***

***massiv.append(element) # Yangi elementni massivga qo'shamiz***

***ko'rsatkich += 1 # Ko'rsatkichni oshiramiz***

***# Misol uchun, yangi elementlarni qo'shish***

***yangi\_element\_qosh(massiv, 10)***

***yangi\_element\_qosh(massiv, 20)***

***yangi\_element\_qosh(massiv, 30)***

***print(massiv) # [10, 20, 30]***

***print(ko'rsatkich) # 3***

Bu yerda massivda 10, 20 va 30 qiymatlari mavjud bo‘lib, ko‘rsatkich 3 ga teng bo‘ladi. Har bir qo‘shish jarayonida ko‘rsatkich yangilanadi. Agar qo‘shimcha savollaringiz bo‘lsa, marhamat!

***Muxtoraliyev Abdulhamid 640-23.***

***2024.***