**Navbatsiz idish. Navbatdagi asosiy operatsiyalar.**

Navbat bu FIFO (First In - First Out - "birinchi kelgan – birinchi ketadi"), shunday o’zgaruvchan uzunlikdagi ketma-ketlik, ro’yhatki, unda tuzilmaga elementlar faqat bir tomondan, ya’ni navbatning oxiridan qo„shiladi va elementlarni tuzilmadan chiqarish boshqa tomondan, ya’ni navbat boshidan amalga oshiriladi. Navbat ustida bajariladigan asosiy amallar

* + yangi elementni qo„shish,
  + elementni chiqarib tashlash,
  + uzunligini aniqlash,
  + navbatni tozalash.

Navbatni statik xotirada vektor ko’rinishida ifodalashda 2 ta parametr, ya’ni navbat boshini (navbatning 1-elementini) va oxirini (navbatning oxirgi elementini) ko„rsatuvchi Ko’rsatkichlar olinadi (1-rasm).

chiqish kirish

Navbat boshi

R=9

Navbat oxiri

1-rasm. Navbat tuzilmasi

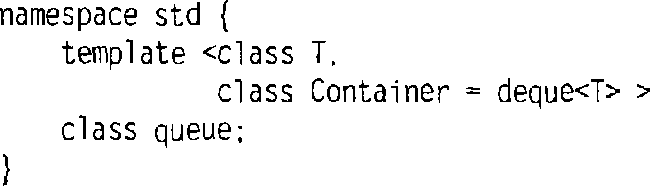
Navbatga yangi element kiritilayotganda navbat oxiri Ko’rsatkichi ko’rsatayotgan adresga yoziladi va shundan keyin navbat oxiri Ko’rsatkichi bittaga oshiriladi. Navbatdan elementni o„chirishda navbat boshi Ko’rsatkichi ko’rsatayotgan adresdagi element o„chiriladi va shundan keyin bu Ko’rsatkichning qiymati bittaga oshiriladi. Navbatga elementlar kiritilganda navbat oxiri Ko’rsatkichi shu navbat uchun ajratilgan xotira sohasining oxiriga yetib qoladi. Bunda navbat to„lgan hisoblanadi.

Agar navbatdan elementlar o„chiriladigan bo’lsa, navbat boshida bo’sh joy ajratiladi. Vaholanki, navbat oxiri Ko’rsatkichi chegaraga yetib qolganligi sababli, navbatga yangi element kiritib bo„lmaydi. Shu sababli navbatda har safar element o„chirilganda qolgan barcha elementlar bitta oldinga surilishi kerak bo’ladi. Natijada navbat oxirida bo’sh joy ochiladi. Bu holatda navbat boshi Ko’rsatkichiga xojat qolmaydi. Lekin shuni aytish kerakki, bu yondashuv bir muncha noqulay hisoblanadi. Shuning uchun har safar elementlarni surib o„tirmaslik uchun navbatni halqasimon shaklda tashkil etamiz. Ya’ni bunda xotirada navbat sohasining oxiriga yetib borilganda navbat boshiga o„tib ketiladi. Ushbu holatda navbat boshi va oxiri Ko’rsatkichlari xotiradagi navbat sohasining boshini ko„rsatadi. Bu ikkala Ko’rsatkichlarning tengligi navbatning bo’shligini anglatadi. Halqasimon navbatda element qo„shish amali o„chirish amalidan ko„proq bajarilsa, navbat oxiri Ko’rsatkichi navbat boshi Ko’rsatkichiga “yetib oladi”. Bu holat navbat to’laligini anglatadi. Halqasimon navbatda elementni o„chirish ikkala Ko’rsatkich ko’rsatayotgan bitta adresda amalga oshiriladi. Bunday navbatning uzunligi boshi va oxiri Ko’rsatkichlari farqi bilan aniqlanadi.

Navbatni dasturda ishlatish uchun <queue> nomli nomli faylni qo’shishi kerak. <queue> faylni quyidagi tarzda aniqlanadi.

#include <queue>

<Queue> faylida navbat navbati quyidagicha tavsiflanadi:



# Navbatni ishlatilishi misoli

queue() sinfning navbatini ishlatishning misoli:

#include <iostream>

#include <queue>

#include <string>

using namespace std;

int main()

{

queue<string>q;

//navbatga uchta element qo'shish

q.push("Assalomu ");

q.push("alaykum ");

q.push("ustoz");

//navbatdan 2 ta elemntni chiqarish va o'chirish

cout<<q.front();

q.pop();

cout<<q.front();

q.pop();

//2 ta element qo'shish

q.push("Volaykum ");

q.push("salom ");

//1ta element o'chirish

q.pop();

//2ta elementni chiqarish va o'chirish

cout<<q.front();

q.pop();

cout<<q.front()<<endl;

q.pop();

//Navbatning elementlar miqdorini chiqarish

cout<<"navbatdagi elementlar soni: "<<q.size()<<endl;

}

Dasturning bajarilish natijasi quyidagiday bo’ladi:

Assalomu alaykum Volaykum salom

navbatdagi elementlar soni: 0

#include <iostream>

#include <queue>

using namespace std;

int main() {

queue <int> q;

cout << "7 ta butun son kiriting: " << endl;

for (int h = 0; h < 7; h++) {

int a;

cin >> a;

q.push(a); // elementlarni navbatga qo'shish

}

cout << endl;

cout << "Navbatdagi birinchi element: " << q.front() << endl; // birinchisini ko'rsatish

//navbat elementi

q.pop(); // elementni navbatdan olib tashlash

cout << "Yangi birinchi element (o'chirilgandan so'ng): " << q.front() << endl;

if (!q.empty()) cout << "Navbat bo'sh emas!"; // navbatning bo'shligini tekshiring (yo'q)

system("pause");

return 0;

}

Dastur natijasi:

7 ta butun son kiriting:

6

4

5

1

3

4

7

Navbatdagi birinchi element: 6

Yangi birinchi element(o’chirilgandan so’ng): 4

Navbat bo’sh emas!

# Operatsiyalar

*Navbat* ::queue()

* Sukutlikda konstruktor.
* Bo’sh navbatni yasaydi.

explicit *navbat* ::queue (const Container& *cont)*

* *cont* elementlari bilan initsializatsiya qilingan navbatni yaratadi*.*
* *Cont* barcha elemetlari navbatga nusxalanadi. size\_type *navbat*: :size () const()
* Joriy elementlarning sonini qaytaradi.
* Navbatdagi yo'q bo’lgan elementlarni tekshirish uchun empty() funktsiyasidan foydalanish tavsiya etiladi, chunki u tezroq ishlashi mumkin.

bool *navbat* ::empty() const

* Navbatning bo’shligini tekshiradi.
* Ekvivalent (lekin tezroq ishlashi mumkin):

*Navbat*::size()==0

void *navbat:*:push (const *value\_type& elem)*

n navbat *elem* parametri elementlari nusxasini qo’yadi, natijada u yangi oxirgi elementga aylanadi.

*value\_type& ochered* :: front()

const *value\_type& navbat* :: front() const

* Ikkala shakllar navbatdagi elementni qaytaradi, ya'ni birinchi navbatda kiritilgan element (navbatdagi boshqa elementlardan keyin).
* Chaqirishdan oldin, navbat kamida bitta elementni (size()> 0) o'z ichiga olishi kerak, aks holda chaqiruv kutilmagan holatga olib keladi.
* Nokonstant navbatlar uchun birinchi shakli murojaatni qaytaradi, bu navbatdagi elementni o'zgartirishga imkon beradi. Bunisi yaxshimi yoki yo'qmi, o'zingni tanlang.

*Value\_type*& *navbat* back()

const *va1ue\_type& navbat*:: back() const

* Ikkala shakllar navbatdagi oxirgi elementni qaytaradi, ya'ni so’nggi navbatda kiritilgan element (navbatdagi boshqa elementlardan keyin).
* Chaqirishdan oldin, navbat kamida bitta elementni (size()> 0) o'z ichiga olishi kerak, aks holda chaqiruv kutilmagan holatga olib keladi.
* Nokonstant navbatlar uchun birinchi shakl murojaatni qaytaradi, bu navbatdagi oxirgi elementni o'zgartirishga imkon beradi. Bunisi yaxshimi yoki yo'qmi, o'zingni tanlang.

void *navbat* :: pop()

* navbatdan navbatdagi elementni, ya'ni birinchi bo'lib kiritilgan elementni (navbatning barcha elementlaridan oldin) o'chiradi.
* Funktsiyaning qaytish qiymati yo'q. Yuqori elementning qiymatini qayta ishlash uchun avvalo front() funktsiyasini chaqirish kerak.
* Chaqirishdan oldin, navbat kamida bitta elementni (size()> 0) o'z ichiga olishi kerak, aks holda chaqiruv kutilmagan holatga olib keladi.
* bool *taqqoslash* (const *navbat& queue 1,* const *navbat*& *queue2)*
* Ikkita bir hil tipli nabatlarning taqqoslash qiymatini qaytaradi.
* *Taqqoslash* parametri *—* keyingi operatsiyalardan birisi:
* operator = operator != operator < operator > operator <= operator >=
* Navbatlar bir-biriga teng hisoblanadi, agar ularda elementlar soni bir xil bo’lsa, agar elementlarning juftligi mos kelsa va bir hil ketma-ketligda bo’lsa (ya'ni tenglik uchun ikkita mos keladigan elementni tekshiruvchi har doim true ni beradi).
* Konteynerlar orasidagi "katta/kichik" nisbati leksik mezonlarga muvofiq tekshiriladi, leksikografiya mezonlari lexicographical\_compare() algoritmining tavsifida ko'rib chiqiladi.

# Topshiriq

1. Navbatda birinchi va oxirgi elementlar o’rni almashtirilsin.
2. Navbat o„rtasidagi element o„chirib tashlansin. Agar navbat elementlari soni toq bo’lsa, bitta element, aks holda ikkita element o„chirilsin.
3. Navbatni juft o’rinda turgan elementlari o„chirilsin.
4. Navbat o„rtasiga '+' belgi joylashtirilsin.
5. Navbat eng kichik elementi topilsin va undan keyin 0 joylashtirilsin.
6. Navbat eng katta elementi topilsin va undan keyin 0 joylashtirilsin.
7. Navbat eng kichik elementi o„chirilsin.
8. Navbatda birinchi elementga teng barcha elementlar o„chirilsin.
9. Navbatda oxirgi elementga teng barcha elementlar o„chirilsin.
10. Navbat eng katta elementi o„chirilsin.
11. Navbat eng kichik elementi topilsin va uning o’rniga 0 joylashtirilsin.

Har bir talaba 4 tadan misol ishlashi kerak!