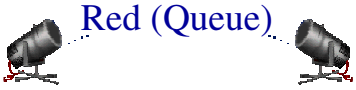
Red (Queue)

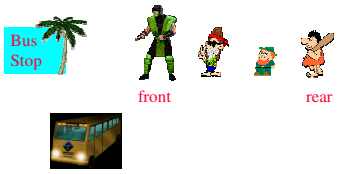
Red (Queue)



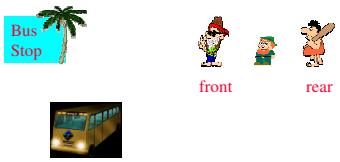
- Linearna lista.
- Jedan kraj liste je front (ili head).
- Drugi kraj liste je rear (ili tail).
- Dodavanje vrijednosti samo sa strane rear.
- Uklanjanje vrijednosti samo sa strane front.

Red za autobus

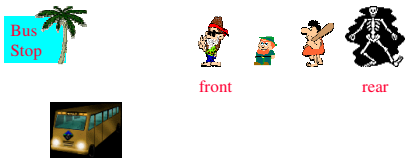
Bus Stop Queue



Bus Stop Queue



Bus Stop Queue

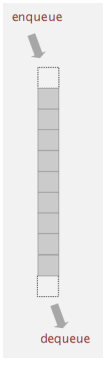


Red (Queue)

- FIFO (First In First Out)
- Prvi element koji se dodaje u strukturu je i prvi element koji napušta strukturu

Operacije sa redom

- IsFullQ … vraća true ako je red pun
- IsEmptyQ … vraća true ako je red prazan
- AddQ … dodaje element na kraj reda (rear) (često se naziva i Enqueue)
- DeleteQ … uklanja i vraća element sa početka reda (front) (često se naziva i Dequeue)

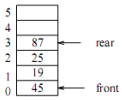


C++ STL implemenatcija reda

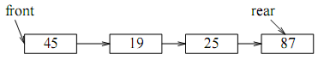
- bool empty()
  - Vraća True ako je red prazan i False u suprotnom.
- T& front()
  - Vraća referencu na vrijednost sa front-a nepraznog reda
- void pop()
  - Uklanja element front-a nepraznog reda.
- void push(const T& foo)
  - Umeće argument foo na kraj reda (rear).
- size_type size()
  - Vraća ukupan broj elemenata u redu.

Implementacija reda

Array/Vector



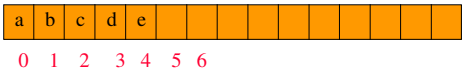
Linked list



Implementacija pomoću niza

- Koristimo 1D niz za predstavljanje reda.
- Elementi se smještaju redom u queue[0], queue[1], itd.

Implementacija pomoću niza



- DeleteQ() => briše queue[0] i pomjera ostale elemente za jednu poziciju ulijevo
  - Vrijeme je O(veličina reda)
- AddQ(x) => ako ima prostora, dodaj lement na desni kraj
  - Vrijeme je O(1)

Brisanje i dodavanje za O(1)

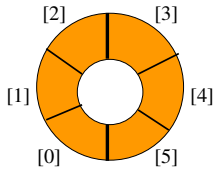
- Da bi se operacije brisanja i dodavanja izvršile za vrijeme O(1), koristimo cirkularnu (kružnu) reprezentaciju.

Cirkularni niz

- Opet koristimo 1D niz za red.

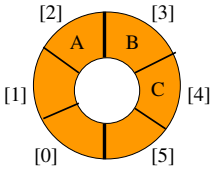
queue[]

- Cirkularni pogled na niz.



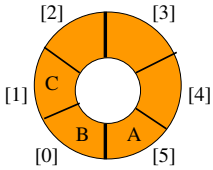
Cirkularni niz

- Moguća konfiguracija sa 3 elementa.



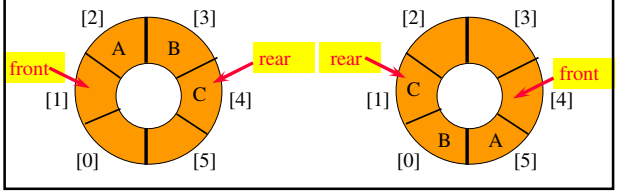
Cirkularni niz

- Druga moguća konfiguracija sa 3 elementa.



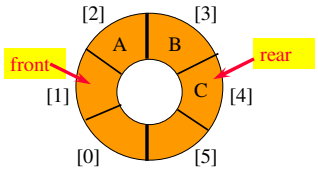
Cirkularni niz

- Koristimo cjelobrojne front i rear.
  - front je jednu poziciju suprotno kazaljki sata od prvog elementa
  - rear daje poziciju posljednjeg elementa



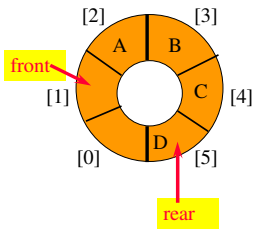
Dodavanje elementa

- Pomjeri se rear jedno mjesto u smjeru kazaljke na satu.



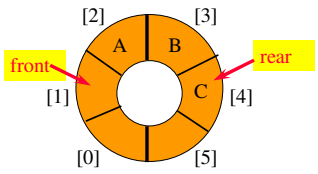
Dodavanje elementa

- Pomjeri se rear jedno mjesto u smjeru kazaljke na satu.
- Zatim se element stavi u queue[rear].



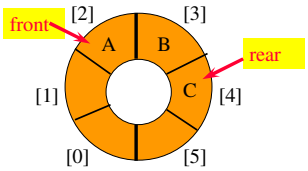
Brisanje elementa

- Pomjeri se front jednu poziciju u smjeru kazaljke sata.



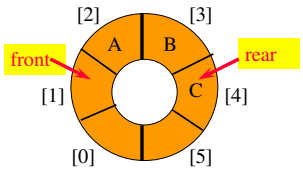
Brisanje elementa

- Pomjeri se front jednu poziciju u smjeru kazaljke sata.
- Zatim se vrati queue[front].



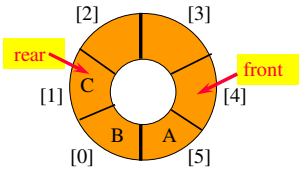
Pomjeranje u smjeru kazaljke sata

- rear++;
  if (rear = = capacity) rear = 0;

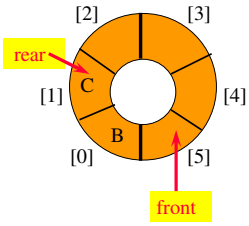


- rear = (rear + 1) % capacity;

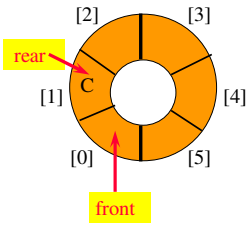
Ispraznimo ovaj red



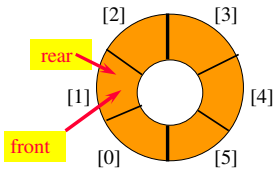
Ispraznimo ovaj red



Ispraznimo ovaj red

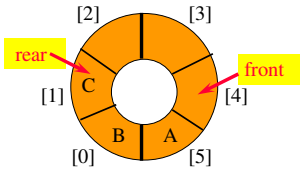


Ispraznimo ovaj red

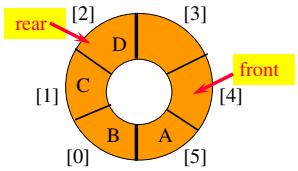


- Kada poslije više brisanja red postaje prazan, imamo front = rear.
- Kada se red konstruiše, on je prazan.
- Inicijalizujemo ga sa front = rear = 0.

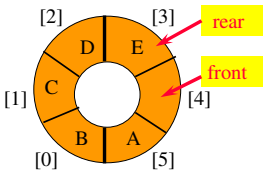
Napunimo red



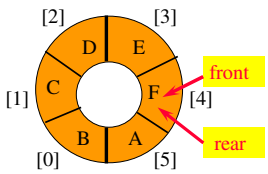
Napunimo red



Napunimo red



Napunimo red



- Poslije niza dodavanja, red je pun, pa je front = rear.
- Ne možemo napraviti razliku između praznog reda i punog reda!

Otklanjanje problema

- Ne dozvoliti da se red napuni.
  - Ako dodavanje elementa uzrokuje prepunjavanje, povećati veličinu niza (moguće u C/C++ i Java-i, nije moguće u Pascal-u).
- Definisati Bulovsku promjenljivu lastOperationIsAddQ.
  - Svako AddQ postavlja ovu promjenljivu na true.
  - Svako DeleteQ postavlja ovu promjenljivu false.
  - Red je prazan ako je (front == rear) && !lastOperationIsAddQ
  - Red je pun ako je (front == rear) && lastOperationIsAddQ

Otklanjanje problema

- Definisati cjelobrojnu promjenljivu size.
  - Svako AddQ odrađuje i size++.
  - Svako DeleteQ odrađuje i size--.
  - Red je prazan ako je (size == 0)
  - Red je pun ako je (size == arrayLength)
- Performanse su neznatno bolje ako se koristi prva startegija.

Implementacija pomoću liste

save item to return

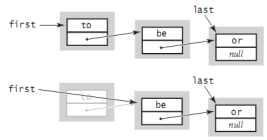
```
String item = first.item;
```

AddQ ili

Enqueue

save item to return

```
first = first.next;
```



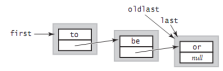
return saved item

```
return item;
```

Implementacija pomoću liste

save a link to the last node

```
Node oldlast = last;
```

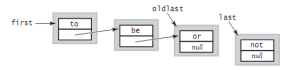


DeleteQ ili

Dequeue

create a new node for the end

```
Node last = new Node();
last.item = "not";
last.next = null;
```



link the new node to the end of the list

```
oldlast.next = last;
```