KOLEJKA (QUEUE)
(lista fifo – first in, first out)

Kolejki są listami, których elementy można wstawiać z jednego końca (rear-tył) a usuwać z drugiego (front - przód).

Operacje:

1. MAKENULL(Q) – czyni kolejkę pustą
2. FRONT(Q) – zwraca pierwszy element kolejki
3. ENQUEUE(x,Q) – wstaw x do kolejki
4. DEQUEUE(Q) – usuwa pierwszy element z kolejki
5. EMPTY(Q) – zwraca true gdy kolejka jest pusta,
   w przeciwnym wypadku zwraca false

Przykład

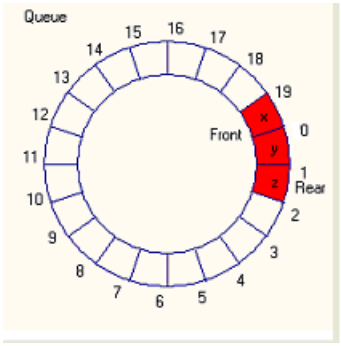
Enqueue(a)
Kolejka: a
Enqueue(b)
Kolejka: a b
Enqueue(c)
Kolejka: a b c

Możemy usunąć tylko element z przodu kolejki (on ma pierwszeństwo, pozostałe elementy muszą czekać na swoją kolej). Tak więc możemy usunąć tylko element *a.*

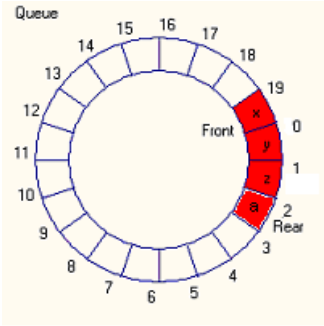
Dequeue()
Kolejka: b c
Dequeue()
Kolejka: c

Tablicowa implementacja kolejki

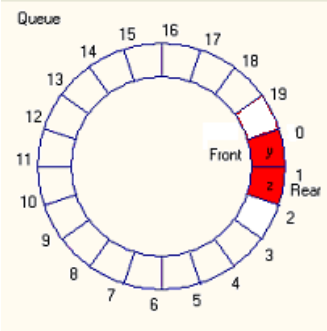
Kolejkę można zaimplementować jako listę zrealizowaną tablicowo, ale jest to bardzo kosztowne.
(usuwając element trzeba przesunąć wszystkie pozostałe elementy).
Efektywniejszym rozwiązaniem jest traktowanie tablicy jako koło, w którym pierwszy element tablicy jest następnym po ostatnim.



Wtedy dołączenie elementu do kolejki powoduje przesunięcie rear o jedną pozycję zgodnie z ruchem wskazówek zegara.

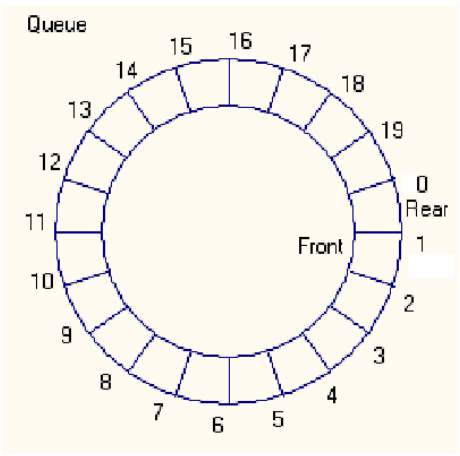


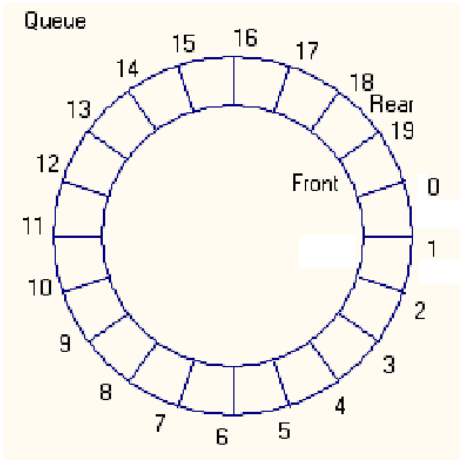
Natomiast usunięcie elementu powoduje przesuniecie front o jedną pozycję zgodnie z ruchem wskazówek zegara.



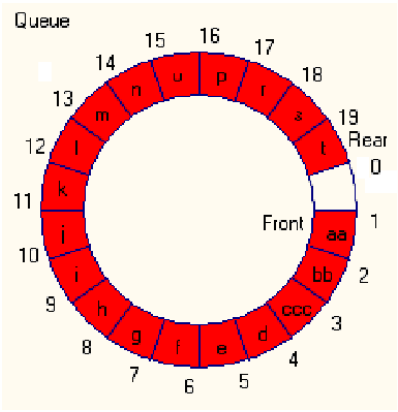
Zakładamy, że kolejka może zawierać tylko maxlen-1 elementów (żeby rozróżnić kolejkę pustą od pełnej)

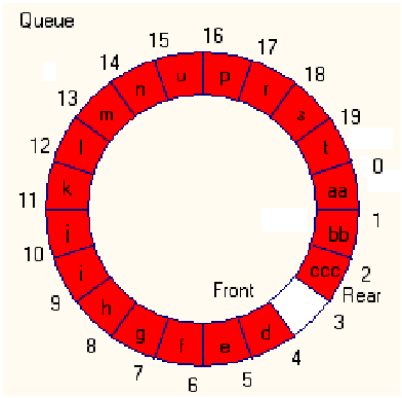
Kolejka jest pusta, jeżeli rear jest “tuż za” front,
na przykład:





Kolejka jest pełna, jeżeli pomiędzy rear a front jest jedena pusta komórka. Na przykład:





Zad.2. Zaimplementować metody klasy Kolejka (reprezentacja cykliczna)

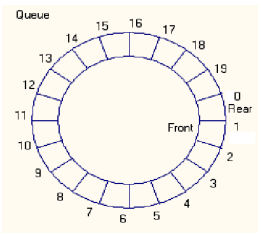
```
const maxlength=50;
typedef int elementtype;
typedef int position;

class Kolejka
{
protected :
     elementtype Queue[maxlength];
     position Front; // Indeks elementu czołowego
     position Rear; // Indeks ostatniego elementu

public:
     Kolejka(){};
     int AddOne(int i);
     void Enqueue(elementtype x);
     void Dequeue();
     elementtype FrontElem();
     void Makenul();
     bool Empty();

};
```

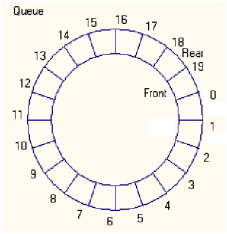
Zad.2. Zaimplementować metody klasy Kolejka (reprezentacja cykliczna)

1) Konstruktor `Kolejka()` powinien inicjować zmienne front i rear tak, aby kolejka była pusta. (np.: przyjmujemy front=0, rear=maxlenght-1)

2) Funkcja pomocnicza `AddOne()` zwiększa o jeden zgodnie z relacją modulo.

3) Operacja EMPTY –zwraca wartość **true**, gdy kolejka jest pusta, zwraca wartość false, gdy nie jest pusta





4) Operacja Enqueue(x) – wstawia element do kolejki (wstawia na koniec !).

-sprawdź, czy jest miejsce w kolejce (czy kolejka nie jest pełna)

Kolejka jest pełna, gdy:
(AddOne(AddOne(Rear)) = Front)

- gdy nie jest pełna, to wstaw element (x) do kolejki

- Zaktualizuj pole rear (pole to wskazuje na ostatnio wstawiony element, chcemy, aby wskazywał na pierwszą wolną komórkę)
- Wstaw elementu x

5) Operacja FrontElem – zwraca pierwszy element z kolejki

- **sprawdź, czy kolejka jest pusta**
- **jeśli nie jest pusta, to zwróć pierwszy element kolejki**
  (element z komórki o numerze Front)

5) Operacja Dequeue – usuń elementu z kolejki. Usuwamy element z początku kolejki!

- sprawdź, czy kolejka jest pusta
- jeśli nie jest pusta, to zmodyfikuj pole Front (zwiększ o 1)