Опашка и дек

1. **Определение**.

Опашката (queue), подобно на стека, представлява свързана, линейна поредица от еднотипни данни. Първият добавен елемент се нарича „начало“, а последният – „край“. От тук се вижда една от основните разлики между опашка и стек – при опашката началото и краят не съвпадат. В резултат на отделянето операциите добавяне и премахване на елемент не се извършват в една и съща точка.

2. **Допустими операции и ограничения.**

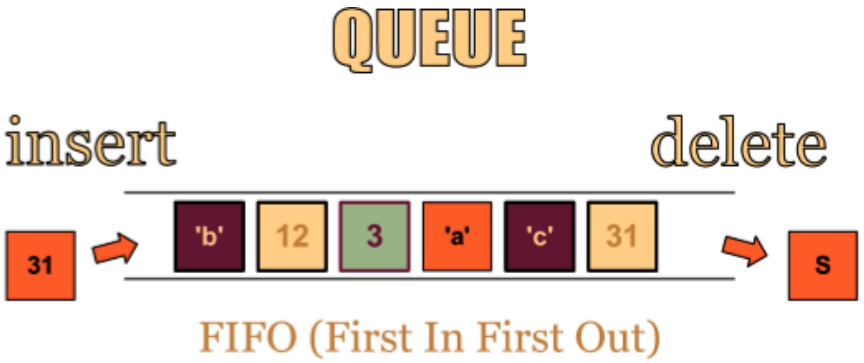
2.1. Добавяне: Нов елемент се добавя само след края на опашката. Така добавеният елемент става последен и съответно се обявява за край на опашката.

2.2. Премахване: Започва се от началото на опашката. След като то се премахне, следващият елемент се обявява за начало.

2.3. Проверка дали опашката е празна.

3. **Особености при опашките.**

Когато в една линейна структура добавянето се извършва в края, а премахването – от началото се получава, че елементите напускат по реда, по който са постъпили (за разлика от стековете). Затова опашките се наричат също и FIFO (First In First Out) списъци.



4. **Програмна реализация.**

Една опашка може да бъде създадена по различни начини:

4.1. Най-бързо става чрез библиотеката „queue“. В нея има шаблонен клас с различни методи подпомагащи реализацията на операции като: добавяне, премахване и проверка за прана опашка, но също и:

4.1.1. Четене на стойността във върха, без той да се премахва: front;

4.1.2. Определяне размера на опашката: size;

4.2. Статична реализация – с едномерен масив.

4.3. Динамична реализация – при нея се използват указатели.

5. **Примерна реализация чрез библиотеката „queue“:**

<u>Пример 1:</u>

```
#include<iostream>
#include<queue>
using namespace std;

int main()
{
   queue<int> qu;
   for(int i=1; i<10; i++) qu.push(i);
```

```
    while(qu.empty() == false)
    {
       cout << qu.front() << " ";
       qu.pop();
    }
    return 0;
}
```

Пример 2:

```
#include <iostream>
#include <queue>
using namespace std;

int main()
{
        queue<int> q;
        cout<<"Broi elementi:"<<q.size()<<endl;
q.push(1);
q.push(2);
q.push(3);
        cout<<"Broi elementi:"<<q.size()<<endl;
        cout<<"Purviq element:"<<q.front()<<endl;
        cout<<"Posledniq element:"<<q.back()<<endl;
q.pop();
        cout<<"Broi elementi:"<<q.size()<<endl;
        cout<<"Purviq element:"<<q.front()<<endl;
        cout<<"Posledniq element:"<<q.back()<<endl;
q.pop();
        cout<<"Broi elementi:"<<q.size()<<endl;
        cout<<"Purviq element:"<<q.front()<<endl;
        cout<<"Posledniq element:"<<q.back()<<endl;
cout<<"-------------------------"<<endl;
queue<char> a;
a.push('a');
a.push('b');
a.push('c');
a.push('d');
a.push('e');
a.push('f');
a.push('g');
        cout<<"Broi elementi:"<<a.size()<<endl;
while(!a.empty())
        {
        a.pop();
        cout<<"Broi elementi:"<<a.size()<<endl;
        }
return 0;
}
```

6. Дек

Дек се нарича списък, при който всички включвания и изключвания на елемент може да бъдат извършвани от двата края на списъка. Идва от английското наименование **Deque**, което пък буквално означава опашка с два края (double-ended queue). Декът е на практика по–рядко използвана структура от данни в сравнение със стека и опашката. В стандартната библиотека на C++ стековете и опашките са реализирани всъщност чрез използване на дек.