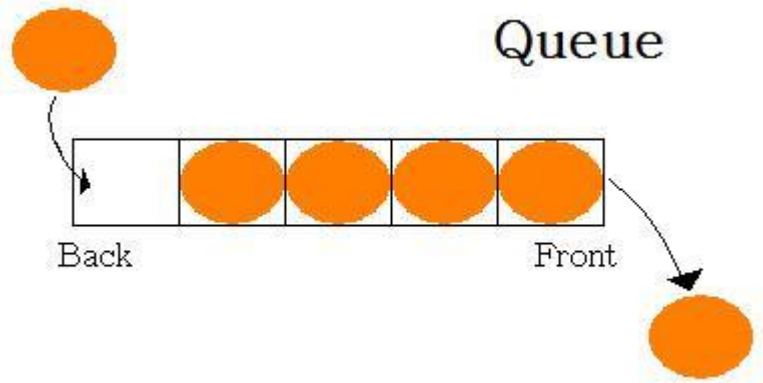
Опашка

доц. д-р Нора Ангелова

Опашка

- Съставна хомогенна линейна структура от данни
- „първи влязъл - пръв излязъл“ (FIFO)



Опашка

Логическо представяне

- Крайна редица от елементи от един и същ тип.

Операции:

- Включване - допустима е само за единия край на опашката (край на опашката).
- Изключване - допустима е само за другия край на опашката (начало на опашката, глава).

Достъп

Възможен е достъп само до елемента, който се намира в началото на опашката.

Достъпът е **пряк**.

Опашка

Операции:

- empty() – проверка дали опашката е празна.
- push(x) – включване на елемент в опашката.
- pop() – изключване на елемент от опашката.
- head() – достъп да първия елемент на опашката.

Опашка

Физическо представяне

- Последователно
- Свързано

Опашка

- Реализации

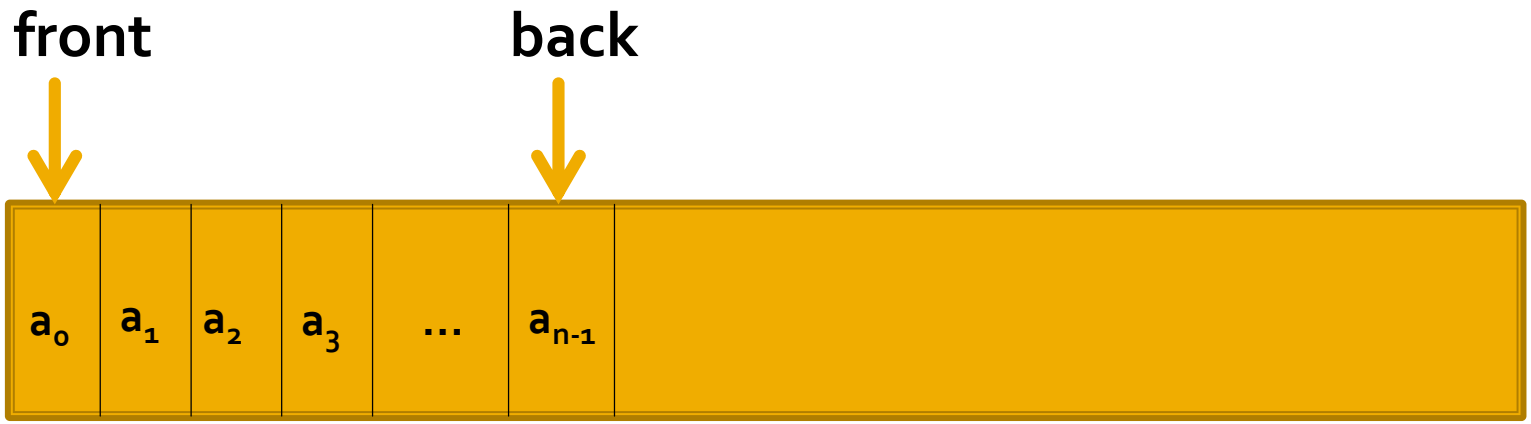
Опашка

Последователно представяне

- Запазва се блок от памет, в който опашката расте и се съкращава.

Опашка - последователно представяне

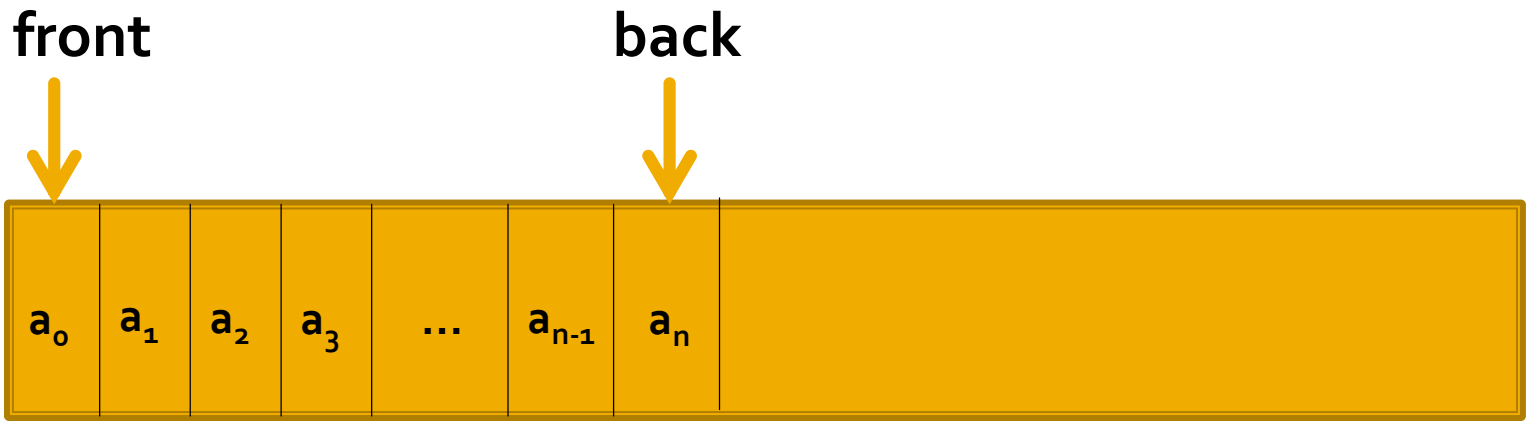
- Масив



Опашка - последователно представяне

Реализация с масив

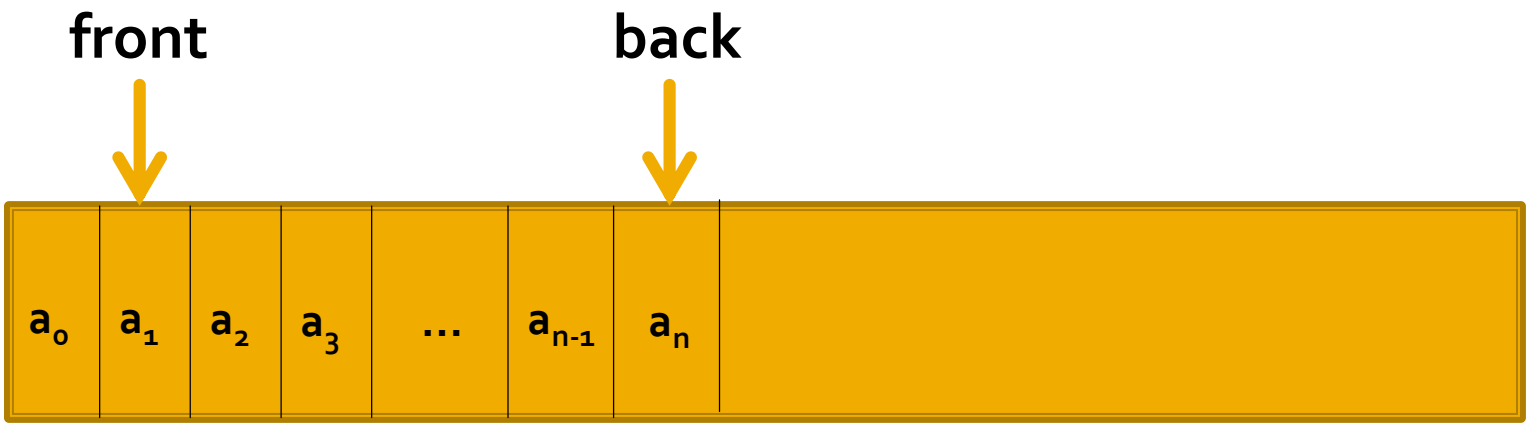
- push ($\mathbf{a_n}$) – включва елемента $\mathbf{a_n}$



Опашка - последователно представяне

Реализация с масив

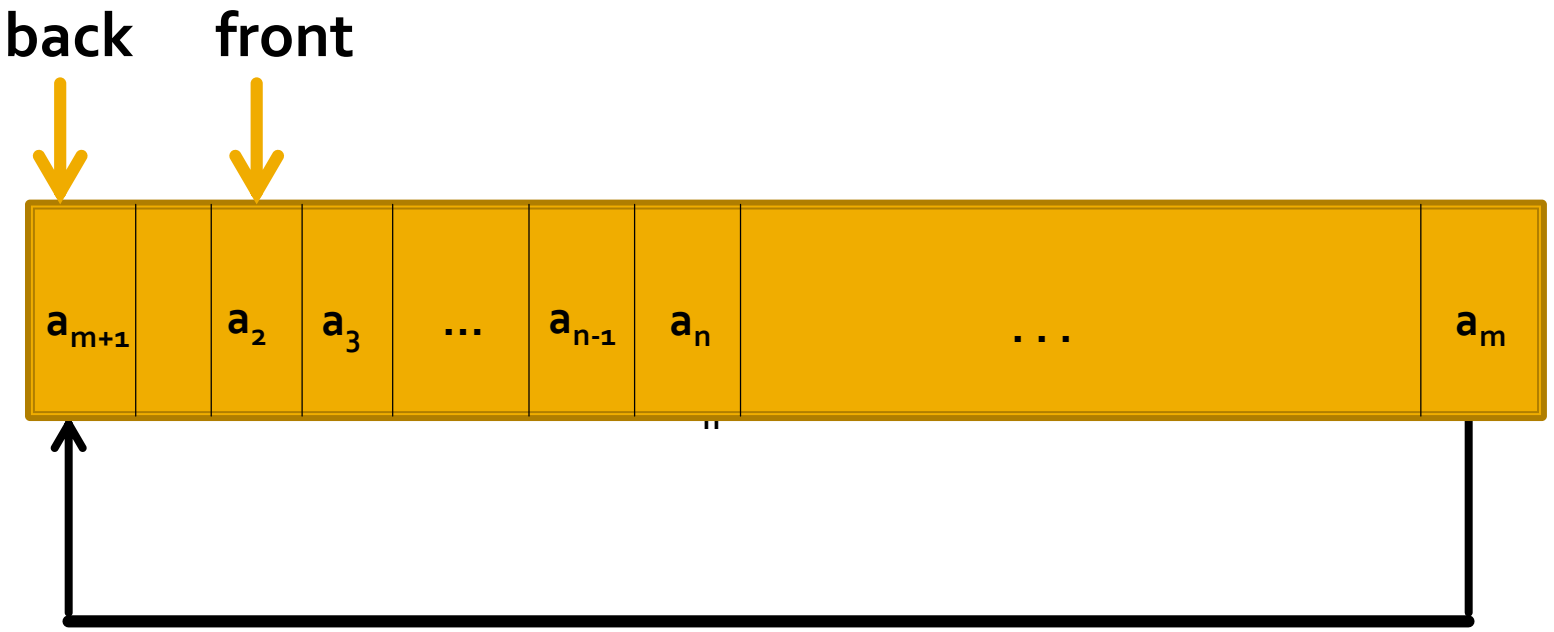
- push ($\mathbf{a_n}$) – включва елемента $\mathbf{a_n}$
- pop () – изключва елемент



Опашка - последователно представяне

Реализация с масив

- push ($\mathbf{a_n}$) – включва елемента $\mathbf{a_n}$
- pop () – изключва елемент
- цикличност



Опашка - последователно представяне

```
const int MAX_SIZE = 100;

template <typename T>
class Queue {
  T arr[MAX_SIZE];
  int front, back, n;      // Индекси и текущ брой на елементите

  bool full() const;

public:
  Queue();                 // Създаване на празна опашка

  bool empty () const;     // Проверка дали опашка е празна
  void push (T const& x); // Включване на елемент
  T pop ();                // Изключване на елемент
  T front() const;         // Достъп до първия елемент в опашка
};
```

Опашка - последователно представяне

```
template <typename T>
// Задава индексите на първата празна позиция
Queue<T>::Queue() : front(0), back(0), n(0) {}

template <typename T>
bool Queue<T>::full() const {
  return n == MAX_SIZE;
}

template <typename T>
bool Queue<T>::empty() const {
  return n == 0;
}
```

Опашка - последователно представяне

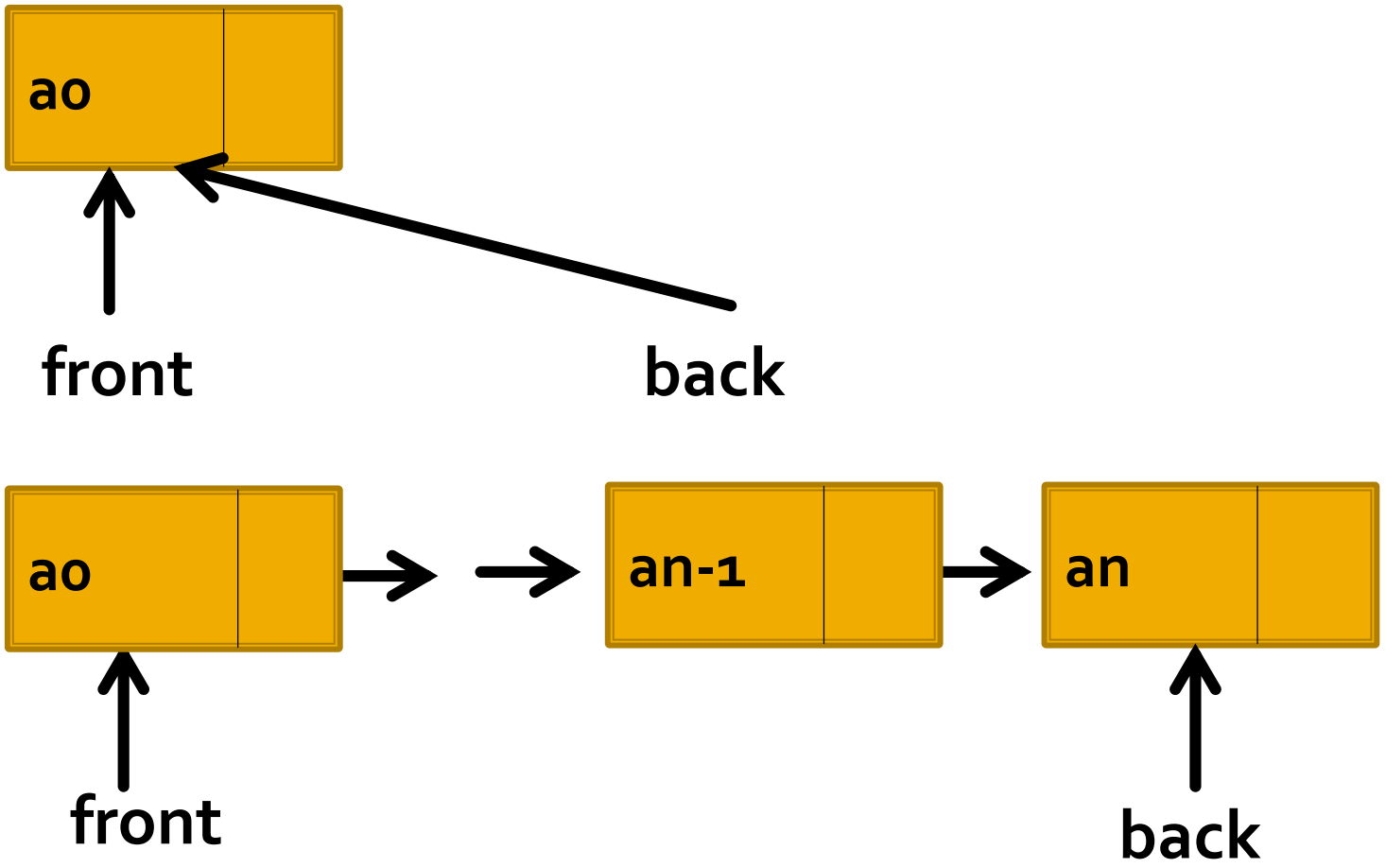
```
template <typename T>
T Queue<T>::front() const {
  if (empty()) {
    std::cerr << "Опашката е празна!\n";
    return T();
  }
  return arr[front];
}

template <typename T>
void Queue<T>::push(T const& x) {
  if (full()) {
    std::cerr << "Опашката е пълна!\n";
    return;
  }

  arr[back] = x;
  n++;
  back++;
  back = back % MAX_SIZE;
}
```

Опашка - последователно представяне

```
template <typename T>
T Queue<T>::pop () {
  if (empty()) {
    std::cerr << "Изкл. на елемент от празна опашка!\n";
    return T();
  }
  T x = arr[front];
  n--;
  front++;
  front = front % MAX_SIZE;
  return x;
}
```

Опашка – свързано представяне

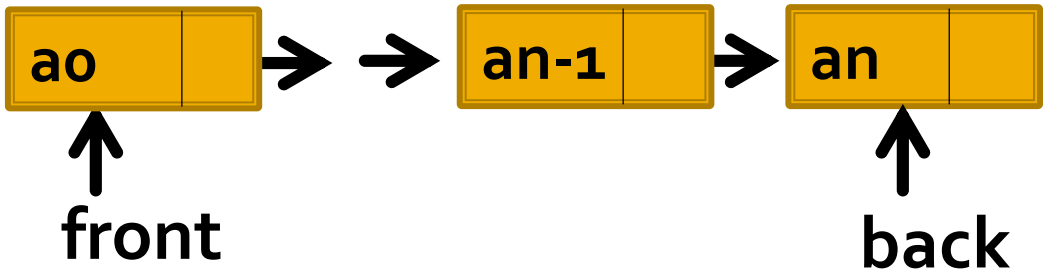
Свързано представяне



Опашка – свързано представяне

Свързано представяне

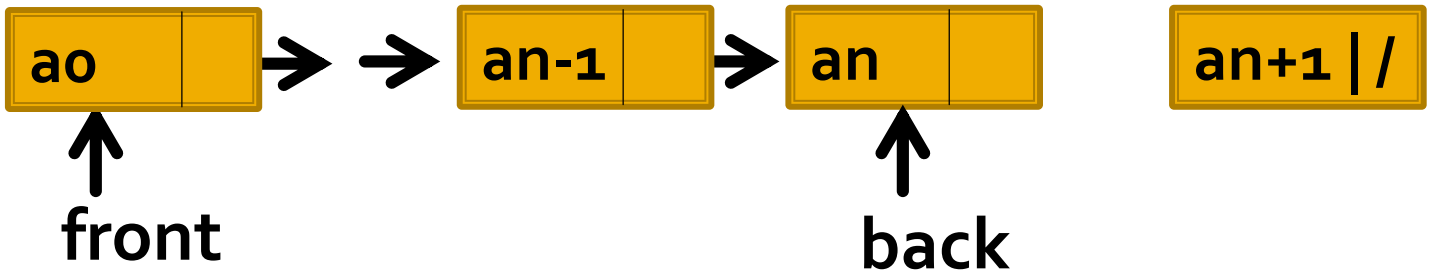
- Добавяне на елемент



Опашка – свързано представяне

Свързано представяне

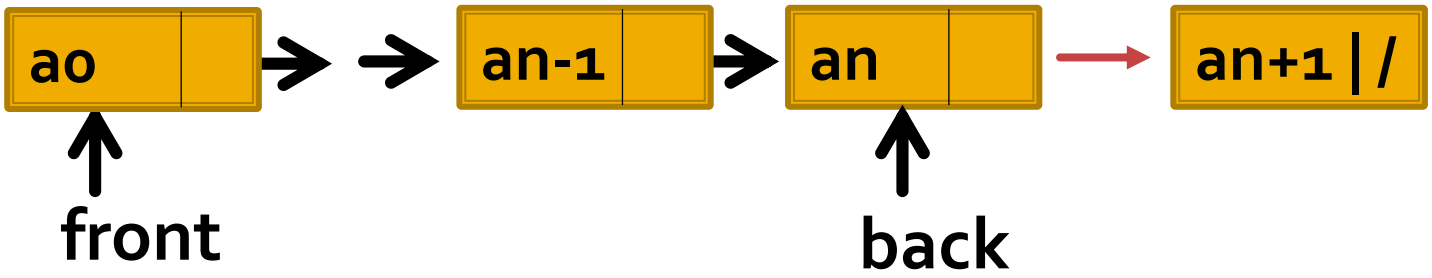
- Добавяне на елемент



Опашка – свързано представяне

Свързано представяне

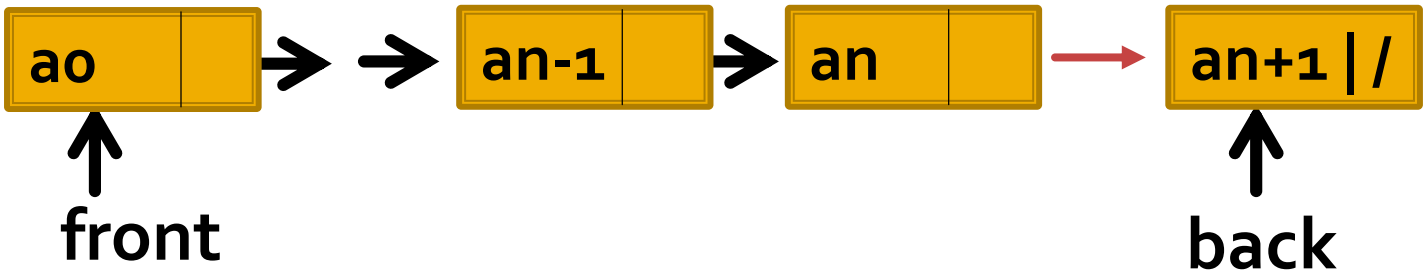
- Добавяне на елемент



Опашка – свързано представяне

Свързано представяне

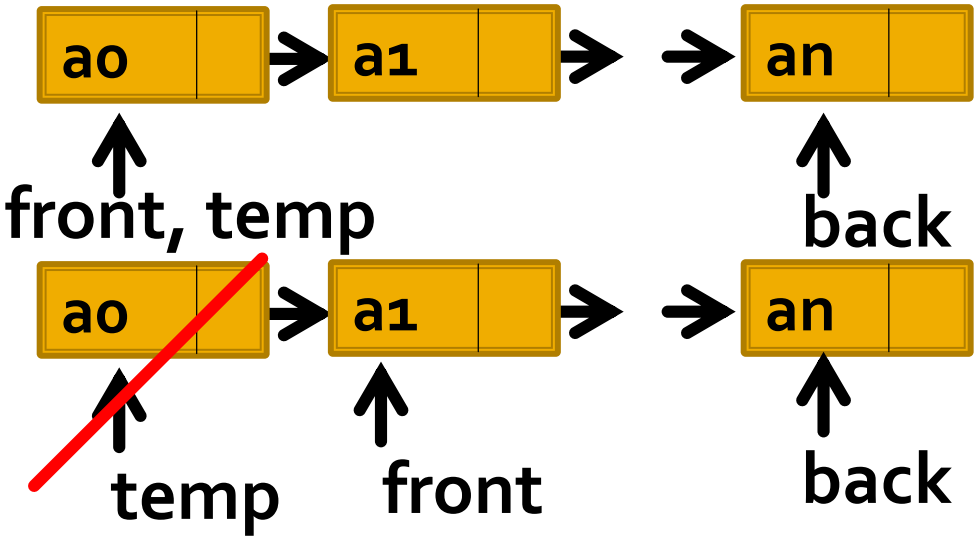
- Добавяне на елемент



Опашка – свързано представяне

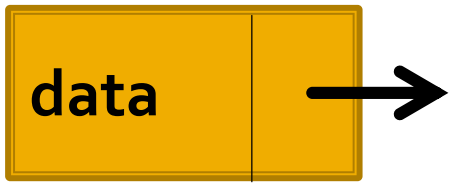
Свързано представяне

- Добавяне на елемент
- Премахване на елемент



Опашка – свързано представяне

```
template <typename T>
struct QueueElement {
  T data;
  QueueElement<T>* next;
};
```



https://github.com/noraAngelova/sdp-kn-2021-2022/tree/main/queue

STL (Опашка)

`std::queue<T>`

`#include <queue>`

Интерфейс:

- queue() — създаване на празна опашка
- empty() — проверка за празнота на опашка
- push(x) — включване на първи елемент в опашката (void)
- pop() — изключване на последен елемент от опашката (void)
- front() — първи елемент в опашката (reference || const_reference)
- back() — последен елемент в опашката (reference || const_reference)
- size() — дължина на опашката

- ==,!=,,<=,>= — лексикорафско сравнение на две опашки

Следва продължение...