Filas

Estrutura de Dados — QXD0010

Prof. Atílio Gomes Luiz
gomes.atilio@ufc.br

Universidade Federal do Ceará

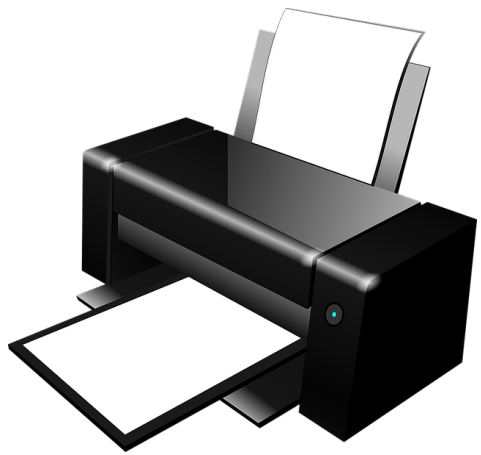
1º semestre/2021

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Introdução

Filas

- Uma impressora é compartilhada em um laboratório
- Alunos enviam documentos quase ao mesmo tempo



Como gerenciar a lista de tarefas de impressão?

Filas

- São `listas lineares` que adotam a política FIFO para a manipulação de elementos.
- FIFO (*first-in first-out*): o primeiro que entra é o primeiro que sai. Remove primeiro objetos inseridos há mais tempo

Filas

- São `listas lineares` que adotam a política FIFO para a manipulação de elementos.
- FIFO (*first-in first-out*): o primeiro que entra é o primeiro que sai. Remove primeiro objetos inseridos há mais tempo

Operações básicas:

- Enfileira (*push_back*): adiciona item no "fim"

Filas

- São `listas lineares` que adotam a política FIFO para a manipulação de elementos.
- FIFO (*first-in first-out*): o primeiro que entra é o primeiro que sai. Remove primeiro objetos inseridos há mais tempo

Operações básicas:

- Enfileira (*push_back*): adiciona item no "fim"
- Desenfileira (*pop_front*): remove item do "início"

Filas

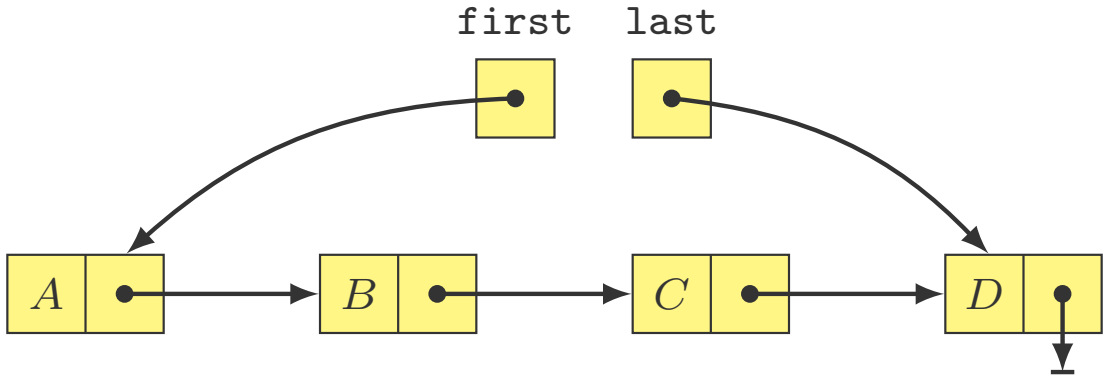
- São `listas lineares` que adotam a política FIFO para a manipulação de elementos.
- FIFO (*first-in first-out*): o primeiro que entra é o primeiro que sai. Remove primeiro objetos inseridos há mais tempo

Operações básicas:

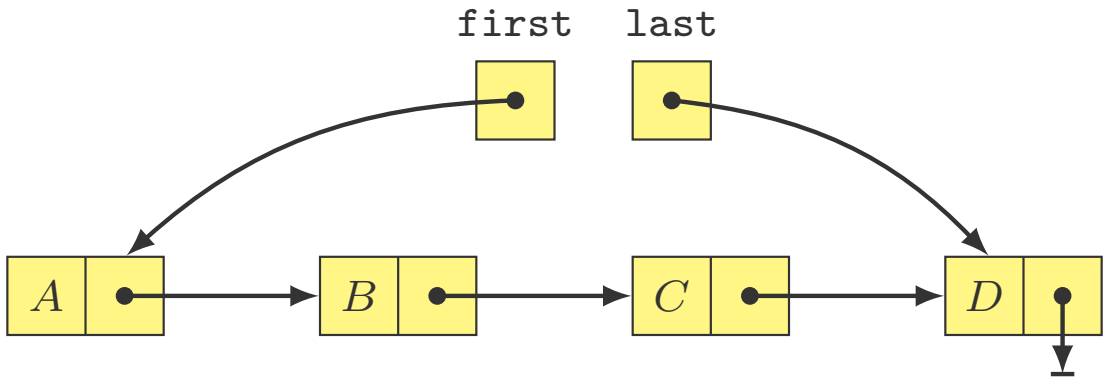
- Enfileira (*push_back*): adiciona item no "fim"
- Desenfileira (*pop_front*): remove item do "início"
- A consulta na fila é feita desenfileirando elemento a elemento até encontrar o elemento desejado ou chegar ao final da fila.

Implementação de uma Fila

Implementação de uma Fila



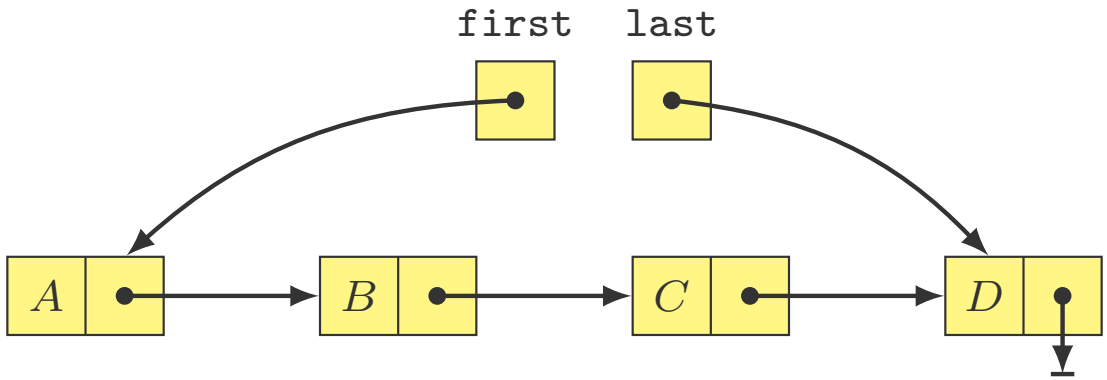
Implementação de uma Fila



- Com relação a alocação de memória, o modo mais natural de implementar uma fila é usando alocação dinâmica.

Implementação de uma Fila

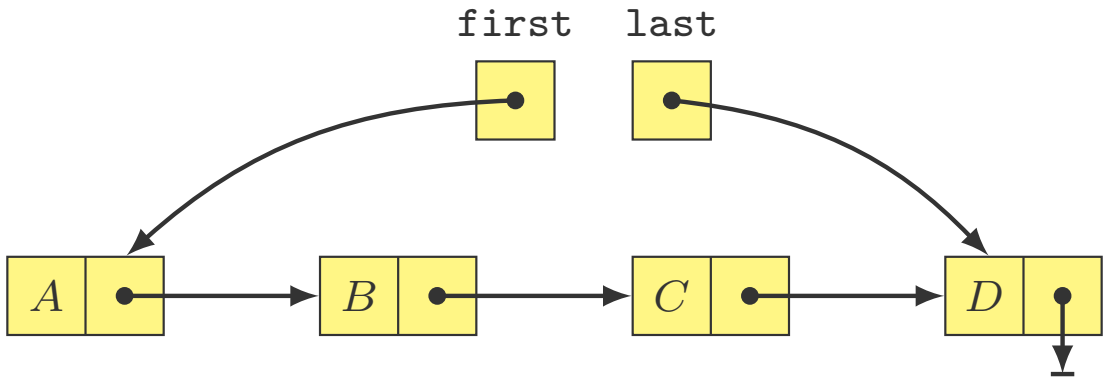
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



- Com relação a alocação de memória, o modo mais natural de implementar uma fila é usando alocação dinâmica.

- Vamos implementar uma fila usando uma lista simplesmente encadeada sem nó cabeça com um ponteiro para o início e outro para o fim da lista.

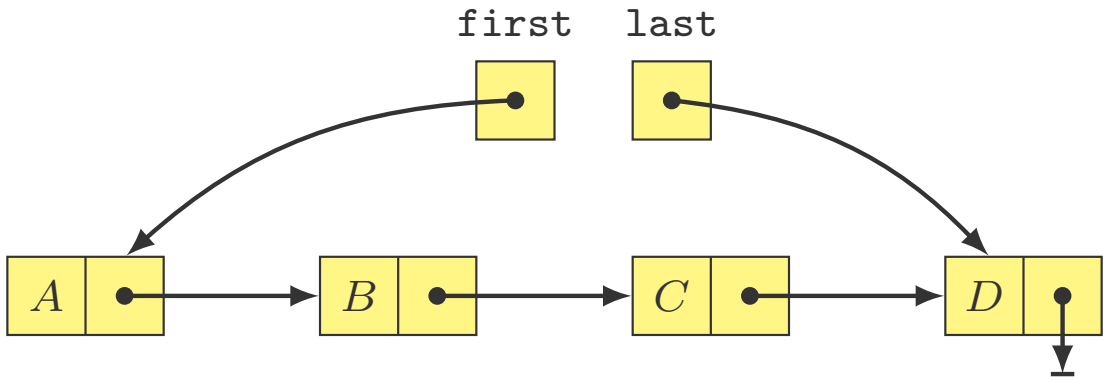
Implementação de uma Fila



- Com relação a alocação de memória, o modo mais natural de implementar uma fila é usando alocação dinâmica.
- Vamos implementar uma fila usando uma lista simplesmente encadeada sem nó cabeça com um ponteiro para o início e outro para o fim da lista.
- Outras variações de lista podem ser usadas:

Implementação de uma Fila

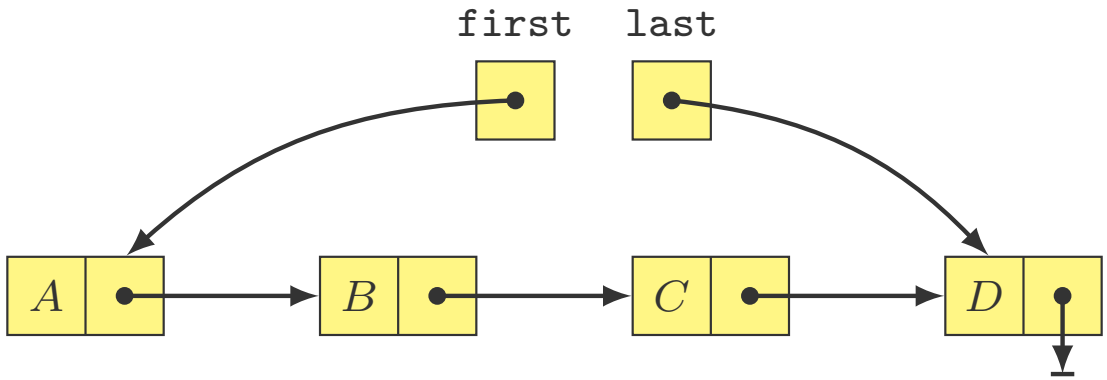
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



- Com relação a alocação de memória, o modo mais natural de implementar uma fila é usando alocação dinâmica.
- Vamos implementar uma fila usando uma lista simplesmente encadeada sem nó cabeça com um ponteiro para o início e outro para o fim da lista.
- Outras variações de lista podem ser usadas:
  - Lista circular simplesmente encadeada;

Implementação de uma Fila

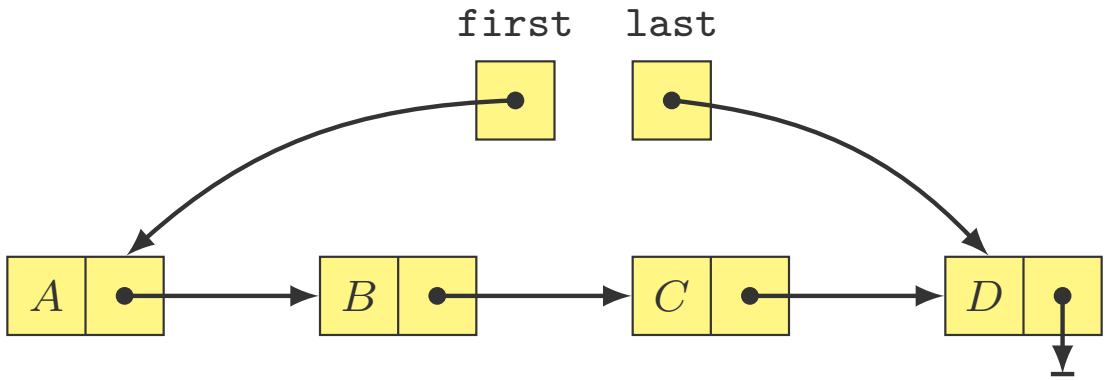
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



- Com relação a alocação de memória, o modo mais natural de implementar uma fila é usando alocação dinâmica.
- Vamos implementar uma fila usando uma lista simplesmente encadeada sem nó cabeça com um ponteiro para o início e outro para o fim da lista.
- Outras variações de lista podem ser usadas:
  - Lista circular simplesmente encadeada;
  - Lista duplamente encadeada;

Implementação de uma Fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



- Com relação a alocação de memória, o modo mais natural de implementar uma fila é usando alocação dinâmica.
- Vamos implementar uma fila usando uma lista simplesmente encadeada sem nó cabeça com um ponteiro para o início e outro para o fim da lista.
- Outras variações de lista podem ser usadas:
  - Lista circular simplesmente encadeada;
  - Lista duplamente encadeada;
  - Lista circular duplamente encadeada, etc.

Implementação de uma Fila

- Nossa fila armazenará números inteiros.

Implementação de uma Fila

- Nossa fila armazenará números inteiros.

- A nível de implementação, cada nó da lista simplesmente encadeada será representado como uma estrutura (struct) que possui apenas dois campos:

Implementação de uma Fila

- Nossa fila armazenará números inteiros.

- A nível de implementação, cada nó da lista simplesmente encadeada será representado como uma estrutura (struct) que possui apenas dois campos:
  - `data`: guarda o valor da chave (um inteiro).

Implementação de uma Fila

- Nossa fila armazenará números inteiros.

- A nível de implementação, cada nó da lista simplesmente encadeada será representado como uma estrutura (struct) que possui apenas dois campos:
  - `data`: guarda o valor da chave (um inteiro).
  - `next`: ponteiro que aponta para o nó seguinte na lista.

Arquivo Node.h


```
1  #ifndef NODE_H
2  #define NODE_H
3
4  typedef int Item;
5
6  // Definição do struct Node
7  struct Node {
8      Item data;  // guarda dado
9      Node *next; // ponteiro para o próximo Node
10
11     Node(const Item& d, Node *ptr) {
12         data = d;
13         next = ptr;
14     }
15 };
16
17 #endif
```

Queue.h — Tipo Abstrato de Dado Fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  #ifndef QUEUE_H
2  #define QUEUE_H
3  #include "Node.h"
4
5  class Queue {
6  private:
7      Node* first; // ponteiro para o primeiro Node
8      Node* last;  // ponteiro para o último Node
9  public:
10     Queue(); // Construtor
11     ~Queue(); // Destrutor
12     bool empty() const; // Lista esta vazia?
13     int size() const; // Devolve o tamanho da fila
14     void push_back(const Item&); // Insere dado no fim
15     void pop_front(); // Remove dado do início
16     Item& front(); // Devolve valor do 1o elemento.
17     Item& back(); // Devolve valor do último elemento.
18 };
19
20 #endif
```

Queue.cpp — Implementação da Fila


```cpp
#include <iostream>
#include <stdexcept>
#include "Node.h"
#include "Queue.h"

// Construtor
Queue::Queue() {
    first = last = nullptr;
}

// Destrutor
Queue::~Queue () {
    while (first != nullptr) {
        Node *temp = first;
        first = first->next;
        delete temp;
    }
}
```

Queue.cpp — Implementação da Fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  // Devolve 'true' se a lista estiver vazia;
2  // e devolve 'false' caso contrario
3  bool Queue::empty() const {
4      return first == nullptr;
5  }
6
7  // Devolve o número de elementos na fila
8  int Queue::size() const {
9      int total = 0;
10     Node *temp = first;
11     while(temp != nullptr) {
12         total++;
13         temp = temp->next;
14     }
15     return total;
16 }
```

Queue.cpp — Implementação da Fila


```
1  // A funçao push_back insere um novo elemento na fila
2  // Cada novo elemento é inserido no final da fila
3  void Queue::push_back(const Item& data) {
4      Node *novo = new Node(data, nullptr);
5      // verifica se fila não está vazia
6      if (last != nullptr)
7          last->next = novo;
8      else    // fila está vazia
9          first = novo;
10     last = novo;
11 }
```

Queue.cpp — Implementação da Fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  // Esta função devolve o último elemento da fila
2  // ou lança uma exceção se a fila estiver vazia.
3  void Queue::pop_front() {
4      if (empty())
5          throw std::overflow_error("erro: fila vazia");
6
7      Node *temp = first;
8      first = temp->next;
9      // verifica se a fila ficou vazia
10     if (first == nullptr)
11         last = nullptr;
12     delete temp;
13 }
```

Queue.cpp — Implementação da Fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  // Devolve uma referência para o primeiro elemento
2  // da lista. Esta operação não remove o elemento.
3  Item& Queue::front() {
4      if (first == nullptr)
5          throw std::overflow_error("erro: fila vazia");
6      return first->data;
7  }
8
9  // Devolve uma referência para o último elemento
10 // da lista. Esta operação não remove o elemento.
11 Item& Queue::back() {
12     if (last == nullptr)
13         throw std::overflow_error("erro: fila vazia");
14     return last->data;
15 }
```

Arquivo main.cpp

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  #include <iostream>
2  #include "Queue.h"  // inclui biblioteca
3  using namespace std;
4
5  int main() {
6      Queue fila; // cria fila vazia
7
8      for(int i = 1; i <=9; i++)
9          fila.push_back(i);  // enfileira
10
11     while(!fila.empty()) {
12         cout << fila.front() << endl;
13         fila.pop_front();
14     }
15
16 }
```

Implementação usando vetor

Fila — Implementação com vetor

Primeira ideia:

- Inserimos no final do vetor: $O(1)$
- Removemos do começo do vetor: $O(n)$

Fila — Implementação com vetor

Primeira ideia:

- Inserimos no final do vetor: $O(1)$
- Removemos do começo do vetor: $O(n)$

Segunda ideia:

- Variável `first` indica o começa da fila
- Variável `last` indica o fim da fila

Fila — Implementação com vetor

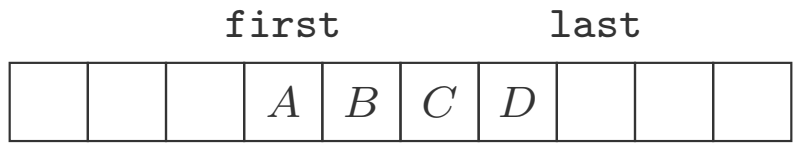
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Primeira ideia:

- Inserimos no final do vetor: $O(1)$
- Removemos do começo do vetor: $O(n)$

Segunda ideia:

- Variável `first` indica o começa da fila
- Variável `last` indica o fim da fila



E se, ao inserir, tivermos espaço apenas à esquerda de `first`?

- podemos mover toda a fila para o começo do vetor
- mas isso leva tempo $O(n)$...

Fila: implementação com vetor (fila circular)

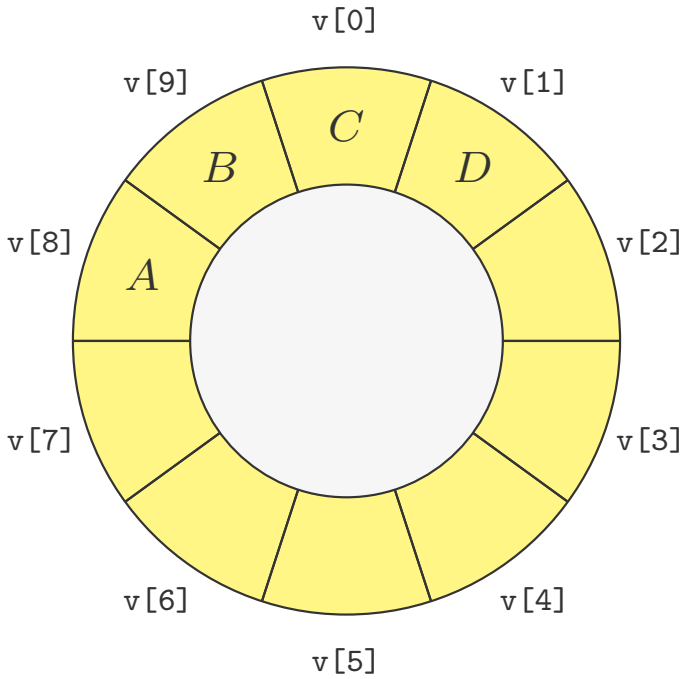
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Solução: considerar o vetor de tamanho `N` de maneira circular

Fila: implementação com vetor (fila circular)

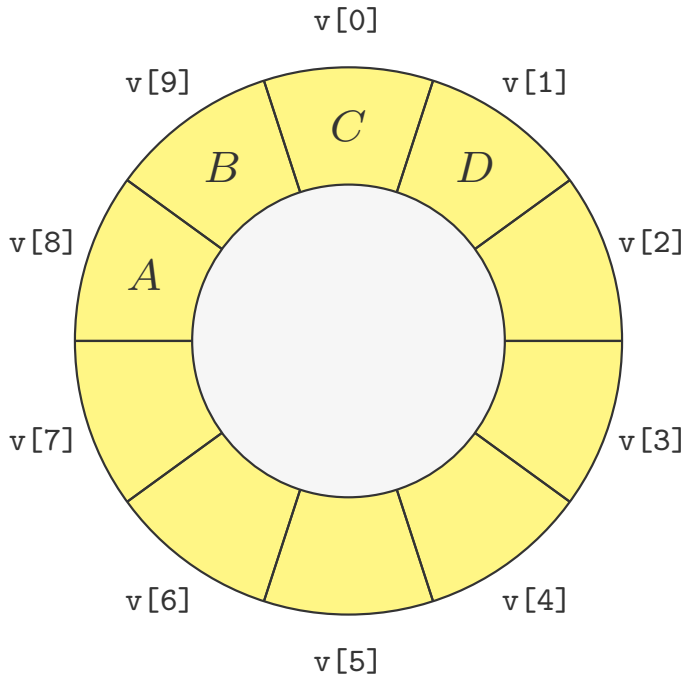
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Solução: considerar o vetor de tamanho `N` de maneira circular



Fila: implementação com vetor (fila circular)

Solução: considerar o vetor de tamanho `N` de maneira circular



As manipulações de índices são realizadas módulo `N`

Queue.h — TAD Fila (implementada com vetor)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  #include <iostream>
2  typedef int Item;
3
4  class Queue {
5  private:
6      Item *array; // ponteiro para vetor de Item
7      int m_size; // Qtd de elementos na Fila
8      int capacity; // Capacidade total da Fila
9      int first; // Indice do primeiro elemento
10 public:
11     Queue(int capacity); // Construtor
12     ~Queue(); // Destrutor
13     void push_back(const Item& data); // Adiciona na fila
14     void pop_front(); // Remove da fila
15     bool empty(); // A fila esta vazia?
16     bool full(); // A fila esta cheia?
17     int size(); // tamanho da fila
18     Item& front(); // Valor do elemento na cabeca
19     Item& back(); // valor do elemento na cauda
20 };
```

Queue.cpp

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
21 #include <iostream>
22 #include <stdexcept>
23 #include "Queue.h"
24 using namespace std;
25
26 // Construtor
27 Queue::Queue(int capacity) {
28     array = new (std::nothrow) int[capacity];
29     if(array == nullptr) {
30         throw runtime_error("erro: memória indisponível");
31     }
32     this->capacity = capacity;
33     m_size = 0;
34     first = 0;
35 }
36
37 // Destrutor
38 Queue::~Queue() {
39     delete[] array;
40 }
```

Queue.cpp


```
41 bool Queue::empty() {
42     return m_size == 0;
43 }
44
45 bool Queue::full() {
46     return m_size == capacity;
47 }
48
49 int Queue::size() {
50     return m_size;
51 }
```

Queue.cpp

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
52 void Queue::push_back(const Item& key) {
53     // fila cheia: capacidade esgotada
54     if (full()) {
55         throw overflow_error("erro: fila cheia");
56     }
57     // insere elemento na proxima posicao livre
58     int fim = (first + m_size) % capacity;
59     array[fim] = key;
60     m_size++;
61 }
62
63 void Queue::pop_front() {
64     if (empty()) {
65         throw overflow_error("erro: fila vazia");
66     }
67     // retira elemento do inicio
68     first = (first + 1) % capacity;
69     m_size--;
70 }
```

Queue.cpp


```
71 Item& Queue::front() {
72     if (empty())
73         throw overflow_error("erro: fila vazia");
74     return array[first];
75 }
76
77 Item& Queue::back() {
78     if (empty())
79         throw overflow_error("erro: fila vazia");
80     return array[(first + m_size-1) % capacity];
81 }
```

main.cpp


```
1  #include <iostream>
2  #include "Queue.h"
3  using namespace std;
4  const int MAX = 10;
5
6  int main() {
7      Queue fila(MAX);
8
9      for (int i = 1; i <= 50; i++)
10         if(!fila.full())
11             fila.push_back(i);  // enfileirando
12
13     while (!fila.empty()) {
14         cout << fila.front() << endl;
15         fila.pop_front();
16     }
17
18 }
```

Exemplos de aplicações de filas

Algumas aplicações de filas:

Exemplos de aplicações de filas

Algumas aplicações de filas:

- Gerenciamento de fila de impressão

Exemplos de aplicações de filas

Algumas aplicações de filas:

- Gerenciamento de fila de impressão
- Buffer do teclado

Exemplos de aplicações de filas

Algumas aplicações de filas:

- Gerenciamento de fila de impressão
- Buffer do teclado
- Escalonamento de processos

Exemplos de aplicações de filas

Algumas aplicações de filas:

- Gerenciamento de fila de impressão
- Buffer do teclado
- Escalonamento de processos
- Comunicação entre aplicativos/computadores

Exemplos de aplicações de filas

Algumas aplicações de filas:

- Gerenciamento de fila de impressão
- Buffer do teclado
- Escalonamento de processos
- Comunicação entre aplicativos/computadores
- Percurso de estruturas de dados complexas (grafos etc.)

Exercícios

Exercício 1 (Filas)

Considere o tipo abstrato de dados Queue como definido nesta aula:

- Sem conhecer a representação interna desse tipo abstrato e usando apenas as operações declaradas no arquivo de interface `Queue.h`, implemente uma função que receba três filas, `f_res`, `f1` e `f2`, e transfira alternadamente os elementos de `f1` e `f2` para `f_res`.

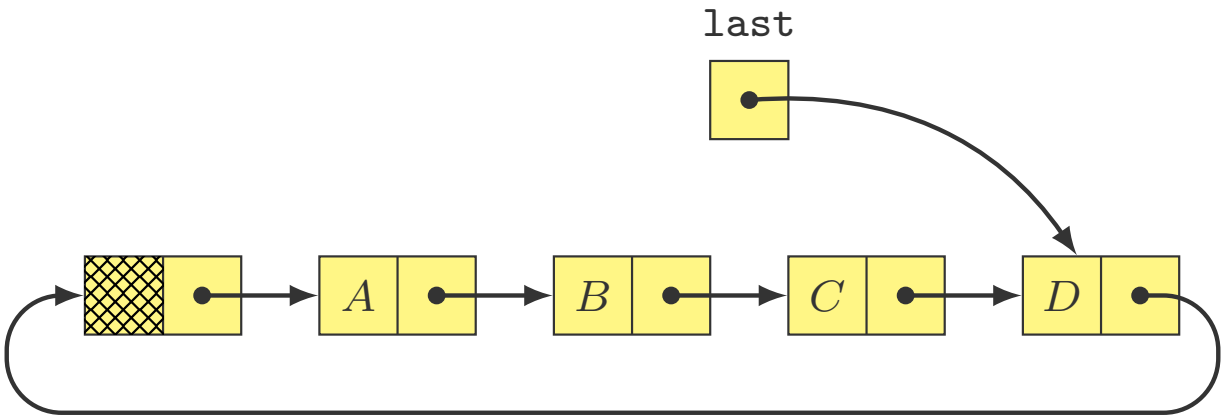
- Note que, ao final dessa função, as filas `f1` e `f2` vão estar vazias, e a fila `f_res` vai conter todos os valores originalmente em `f1` e `f2` (inicialmente `f_res` pode ou não estar vazia).

- Essa função deve obedecer ao protótipo:
  `void combina_filas(Queue *f_res, Queue *f1, Queue *f2)`

Exercício 2 — Implementação Alternativa (Filas)

Universidade Federal do Ceará
Campus Quixadá

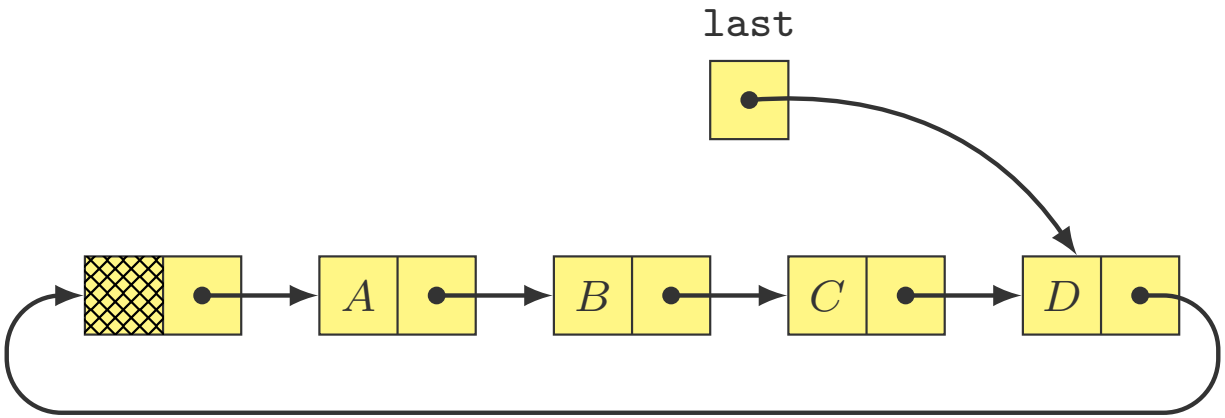
Exercício: implemente uma fila em uma lista encadeada circular com nó cabeça.



Exercício 2 — Implementação Alternativa (Filas)

Universidade Federal do Ceará
Campus Quixadá

Exercício: implemente uma fila em uma lista encadeada circular com nó cabeça.



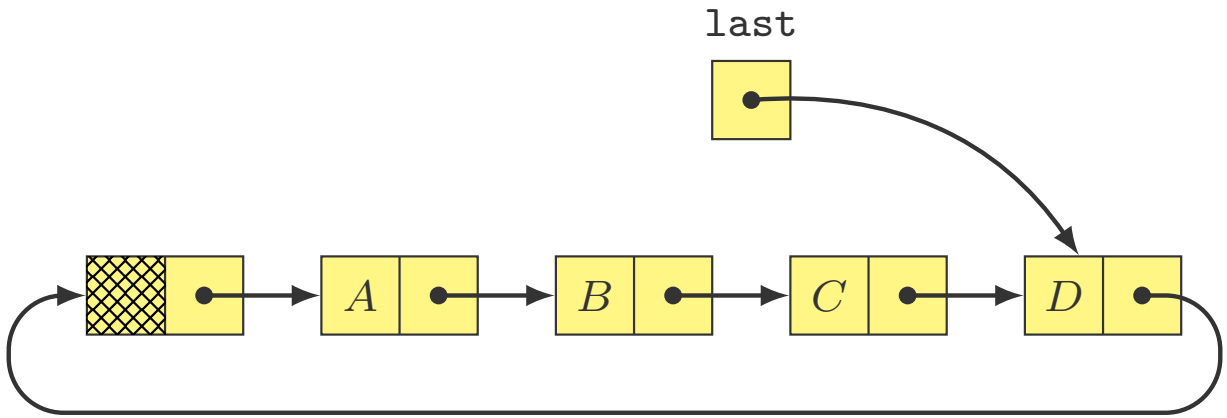
Enfileira:

- Atualizar o campo `next` de `last`
- Mudar `last` para apontar para o novo nó

Exercício 2 — Implementação Alternativa (Filas)

Universidade Federal do Ceará
Campus Quixadá

Exercício: implemente uma fila em uma lista encadeada circular com nó cabeça.



Enfileira:

- Atualizar o campo `next` de `last`
- Mudar `last` para apontar para o novo nó

Desenfileira:

- Basta remover o nó seguinte ao nó auxiliar
  - isto é, `last->next->next`

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

FIM