Filas

Estrutura de Dados — QXD0010

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

Universidade Federal do Ceará

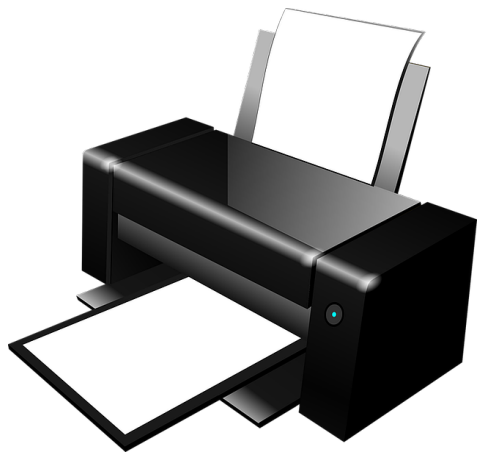
1º semestre/2021

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 (*queue*): 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 (*queue*): adiciona item no "fim"
- Desenfileira (*dequeue*): remove item do "início"

Filas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

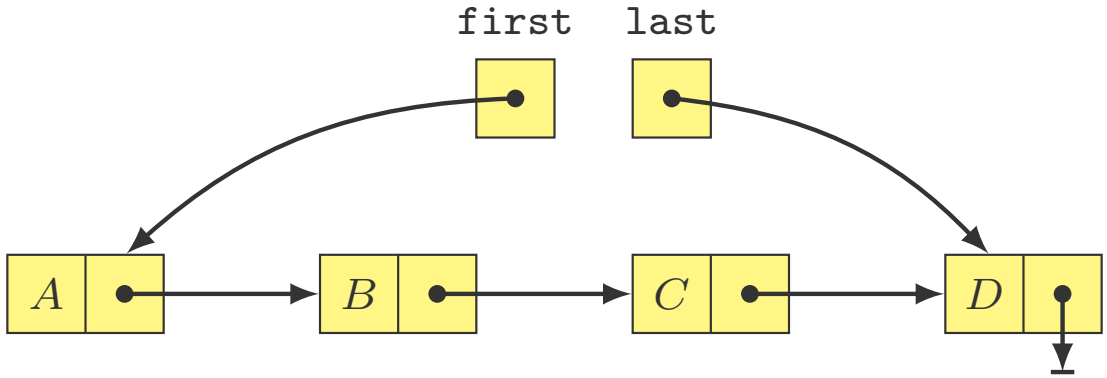
- São `listas lineares` que adotam a política FIFO para a manipulação de elementos.
- FIFO (*first-in first-out*): o primeiro que entra é o primeiro que sai. Remove primeiro objetos inseridos há mais tempo

Operações básicas:

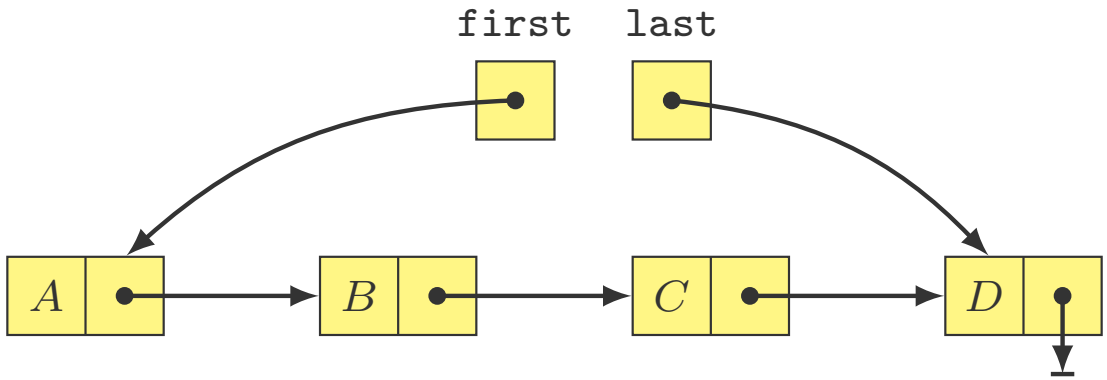
- Enfileira (*queue*): adiciona item no "fim"
- Desenfileira (*dequeue*): remove item do "início"
- A consulta na fila é feita desenfileirando elemento a elemento até encontrar o elemento desejado ou chegar ao final da fila.

Implementação de uma Fila

Implementação de uma Fila



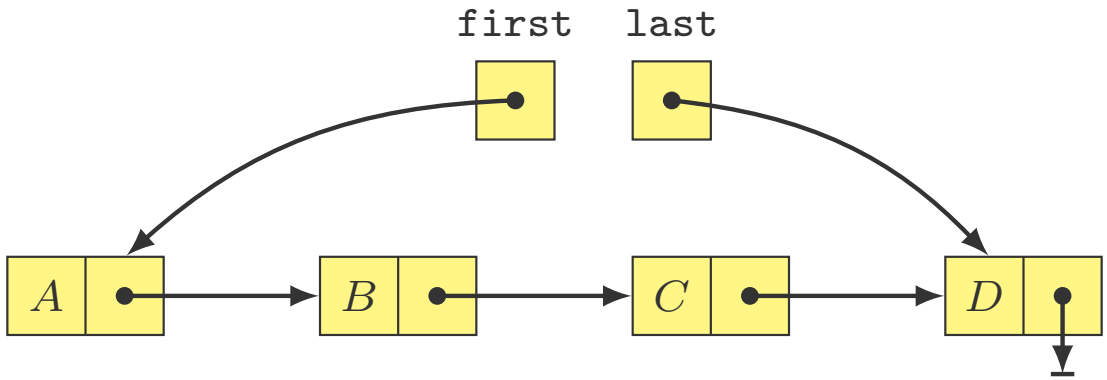
Implementação de uma Fila



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

Implementação de uma Fila

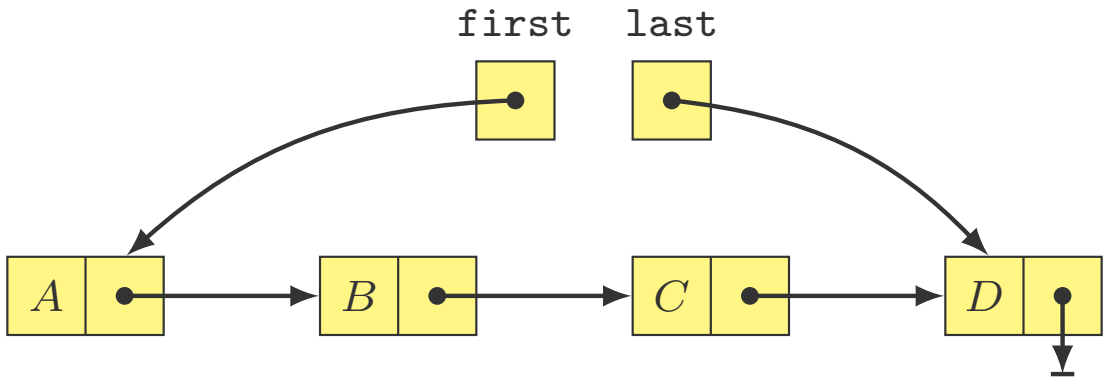
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



- 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.

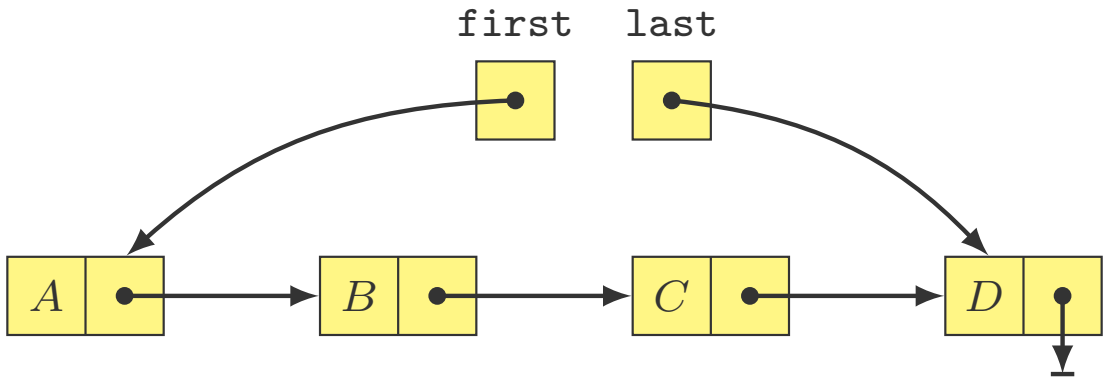
Implementação de uma Fila



- 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

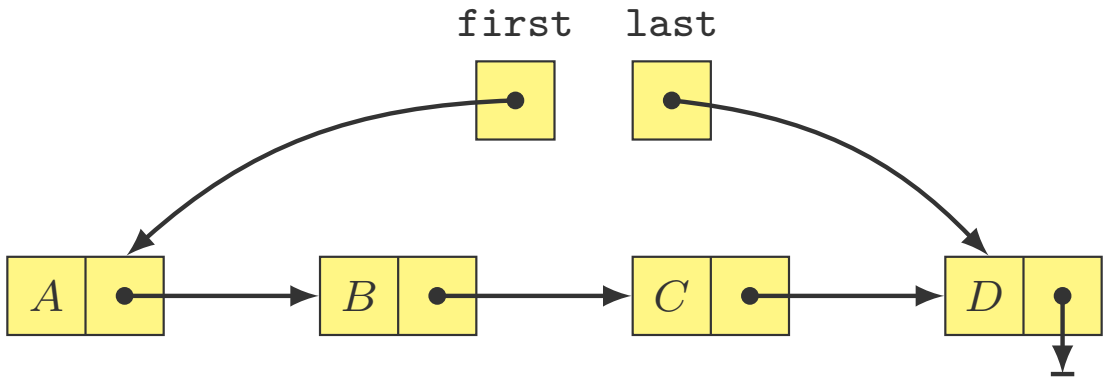
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



- 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

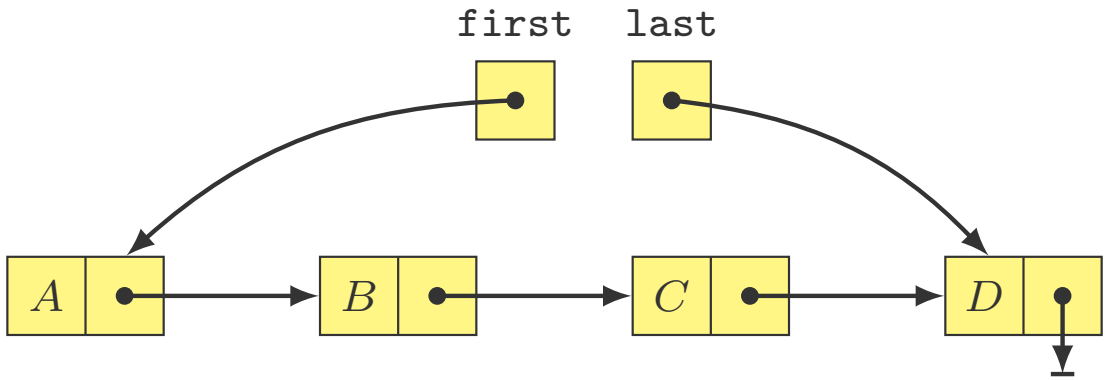
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



- 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:
  - `key`: 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:
  - `key`: guarda o valor da chave (um inteiro).
  - `next`: ponteiro que aponta para o nó seguinte na lista.

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:
  - `key`: guarda o valor da chave (um inteiro).
  - `next`: ponteiro que aponta para o nó seguinte na lista.

- Definição do nó da lista:

```
1 // Definicao do struct Node
2 struct Node {
3     int key;          // guarda inteiro (chave)
4     Node *next;       // ponteiro para proximo no
5 };
```

Queue.h — Tipo Abstrato de Dado Fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  #ifndef QUEUE_H
2  #define QUEUE_H
3
4  struct Node;
5
6  class Queue {
7  private:
8      Node* first; // ponteiro para o primeiro no
9      Node* last;  // ponteiro para o ultimo no
10 public:
11     Queue(); // Construtor
12     ~Queue(); // Destrutor
13     bool empty(); // Lista esta vazia?
14     void enqueue(int key); // Insere key ao final
15     int dequeue(); // Remove primeiro elemento
16     int head(); // Devolve valor do 1o elemento.
17     int tail(); // Devolve valor do ultimo elemento.
18 };
19
20 #endif
```

`Queue.cpp` — Implementação

- As operações da Fila são implementadas no arquivo `Queue.cpp`.

Queue.cpp — Implementação

- As operações da Fila são implementadas no arquivo `Queue.cpp`.
- O nó da lista (`struct Node`) é definido nesse arquivo:

```
1  #include <iostream>
2  #include <climits>
3  #include "Queue.h"
4  using namespace std;
5
6  // Definicao do struct Node
7  struct Node {
8      int key;        // guarda inteiro (chave)
9      Node *next;     // ponteiro para proximo no
10 };
```

- As operações de fila são apresentadas adiante.

Queue.cpp — Implementação da Fila

Construtor e Destrutor:

```
1 Queue::Queue() {          // Construtor
2     first = last = nullptr;
3 }
```

Queue.cpp — Implementação da Fila

Construtor e Destrutor:

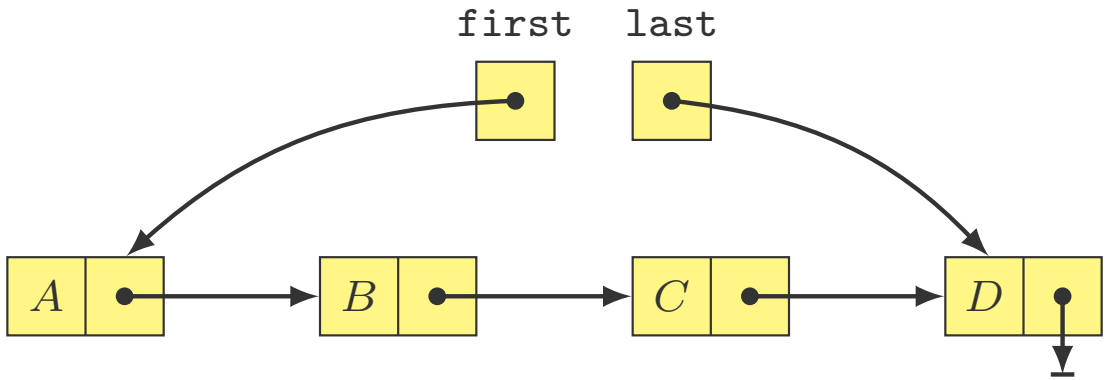
```
1 Queue::Queue() {          // Construtor
2     first = last = nullptr;
3 }
```

Queue.cpp — Implementação da Fila

Construtor e Destrutor:

```
1  Queue::Queue() {          // Construtor
2      first = last = nullptr;
3  }
4
5  Queue::~Queue () {        // Destrutor
6      while (first != nullptr) {
7          Node *aux = first;
8          first = first->next;
9          cout << "Removendo chave " << aux->key << endl;
10         delete aux;
11     }
12     first = last = nullptr;
13 }
```

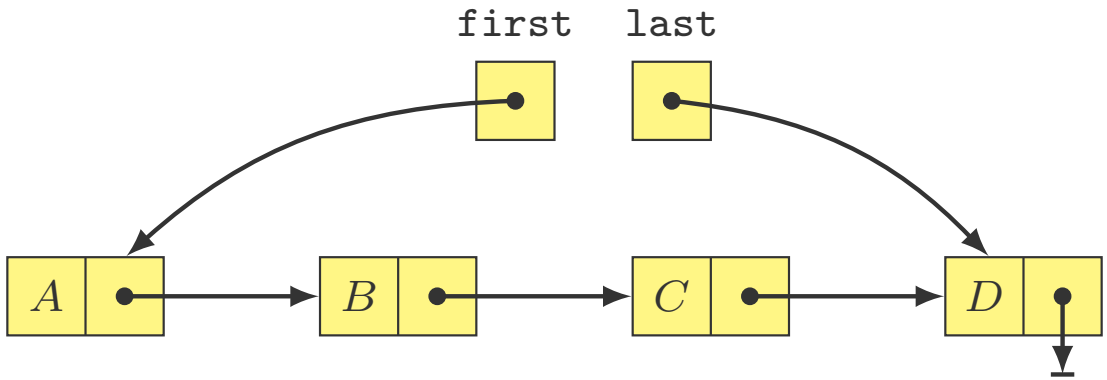
Queue.cpp — Implementação da Fila



Operação `empty()`: saber se a fila está vazia:

```
1 bool Queue::empty() {
```

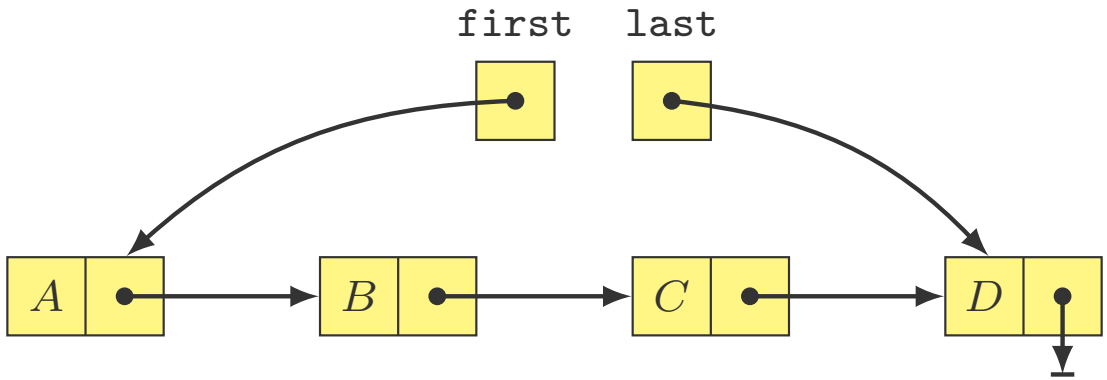
Queue.cpp — Implementação da Fila



Operação `empty()`: saber se a fila está vazia:

```
1 bool Queue::empty() {
```

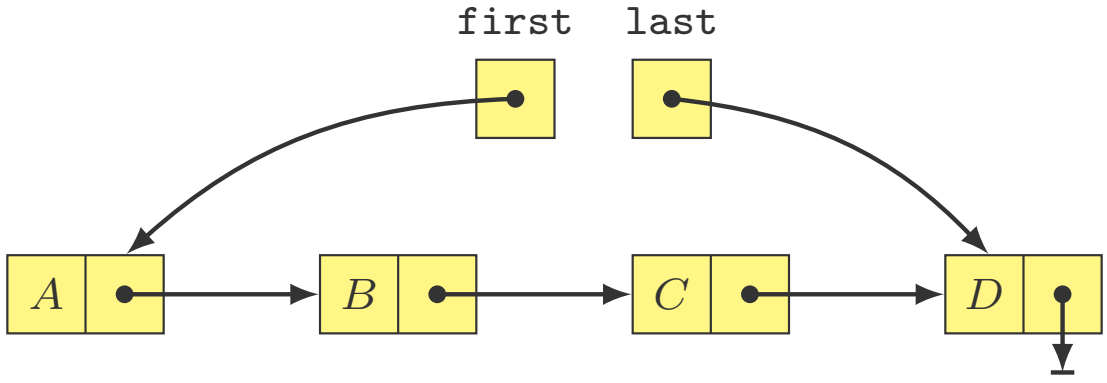
Queue.cpp — Implementação da Fila



Operação `empty()`: saber se a fila está vazia:

```
1 bool Queue::empty() {
2     return (first == nullptr);
3 }
```

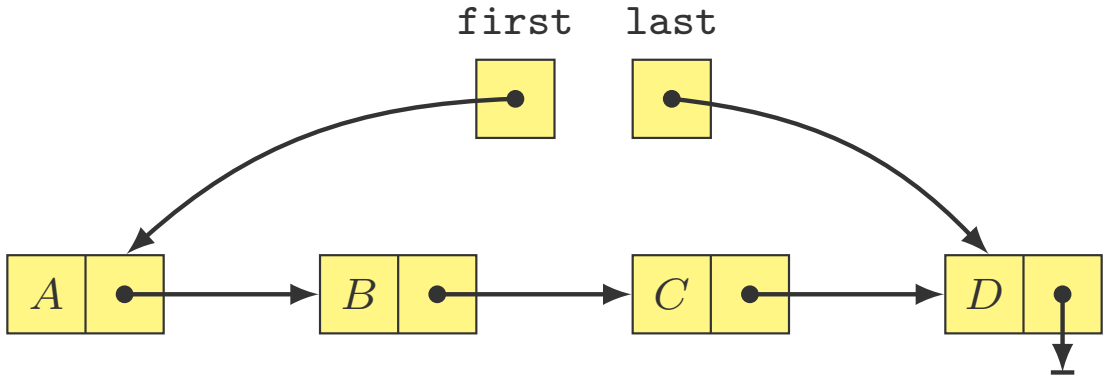
Queue.cpp — Implementação da Fila



Operação `enqueue()`: Insere na fila.

```
1 void Queue::enqueue(int key) {
```

Queue.cpp — Implementação da Fila

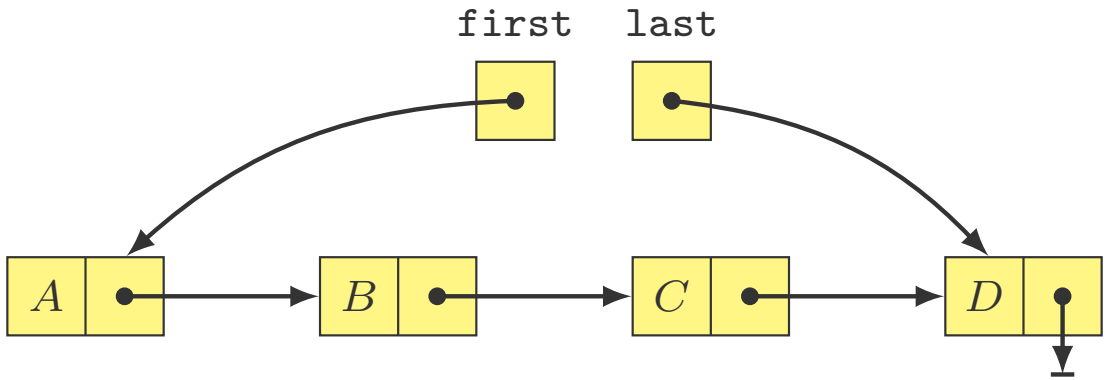


Operação `enqueue()`: Insere na fila.

```
1 void Queue::enqueue(int key) {
```

Queue.cpp — Implementação da Fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

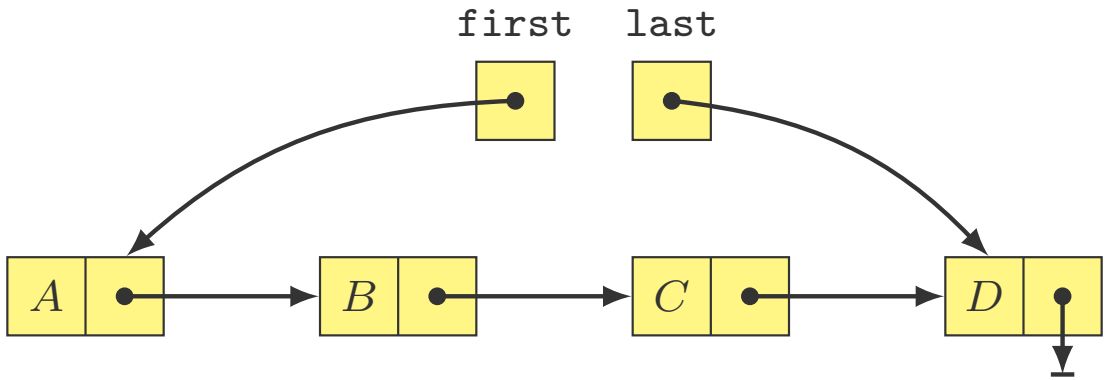


Operação `enqueue()`: Insere na fila.

```
1 void Queue::enqueue(int key) {
2     Node *novo = new Node;
3     novo->key = key;
4     novo->next = nullptr;  // novo node passa a ser o ultimo
```

Queue.cpp — Implementação da Fila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



Operação `enqueue()`: Insere na fila.

```
1  void Queue::enqueue(int key) {
2      Node *novo = new Node;
3      novo->key = key;
4      novo->next = nullptr;  // novo node passa a ser o ultimo
5      if (last != nullptr) // verifica se fila nao estava vazia
6          last->next = novo;
7      else     // fila estava vazia
8          first = novo;
9      last = novo;
10 }
```

Queue.cpp — Implementação da Fila

Operação `dequeue()`: Remove da fila.

```
1 int Queue::dequeue() {
```

Queue.cpp — Implementação da Fila

Operação `dequeue()`: Remove da fila.

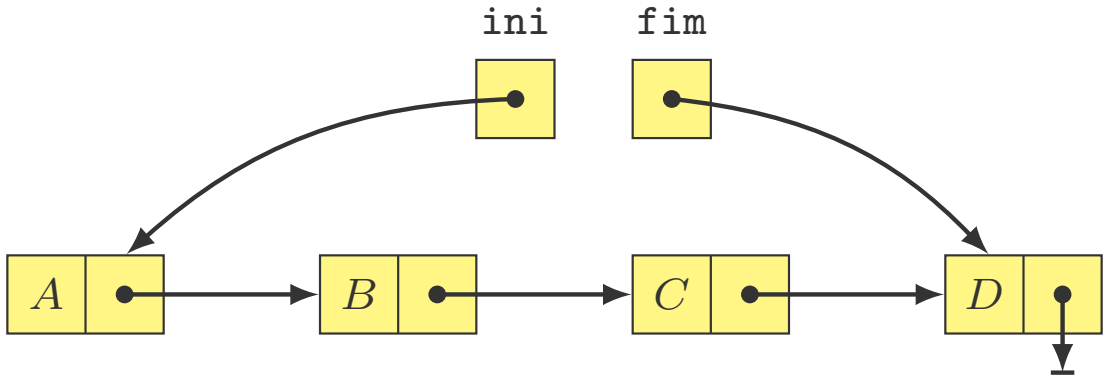
```
1 int Queue::dequeue() {
2     if (empty()) {
3         cout << ">> erro: fila vazia" << endl;
4         return INT_MIN;
5     }
```

Queue.cpp — Implementação da Fila

Operação `dequeue()`: Remove da fila.

```
1  int Queue::dequeue() {
2      if (empty()) {
3          cout << ">> erro: fila vazia" << endl;
4          return INT_MIN;
5      }
6      int key = first->key;
7      Node *aux = first;
8      first = aux->next;
9      // verifica se a fila ficou vazia
10     if (first == nullptr)
11         last = nullptr;
12     delete aux;
13     return key;
14 }
```

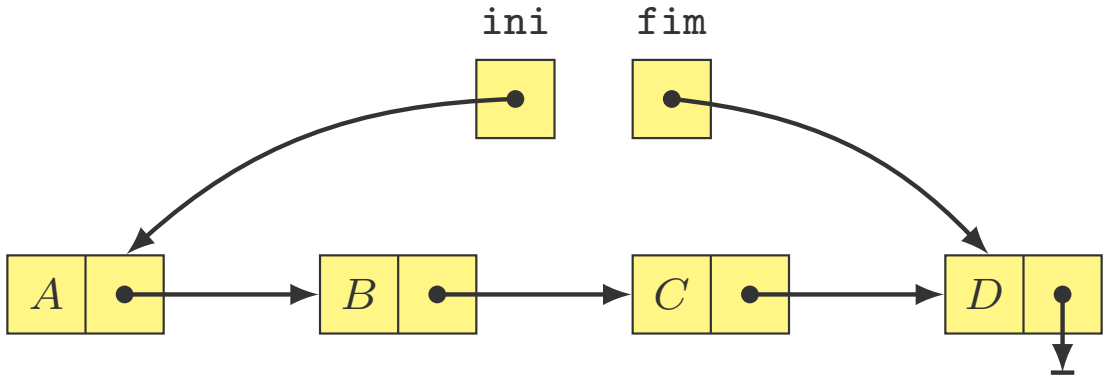
Queue.cpp — Implementação da Fila



Operação `head()`: Consulta elemento na cabeça da lista.

```
1 int Queue::head() {
```

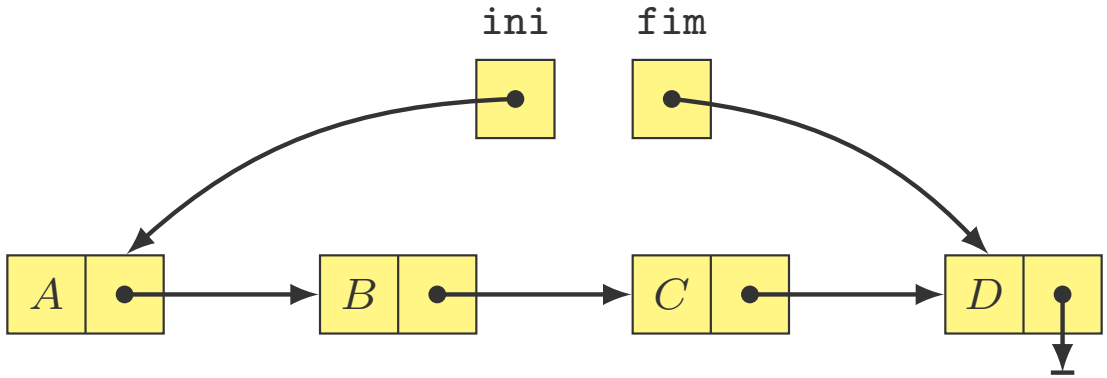
Queue.cpp — Implementação da Fila



Operação `head()`: Consulta elemento na cabeça da lista.

```
1 int Queue::head() {
2     if (first != nullptr)
3         return first->key;
4
5     cout << ">> erro: fila vazia" << endl;
6     return INT_MIN;
7 }
```

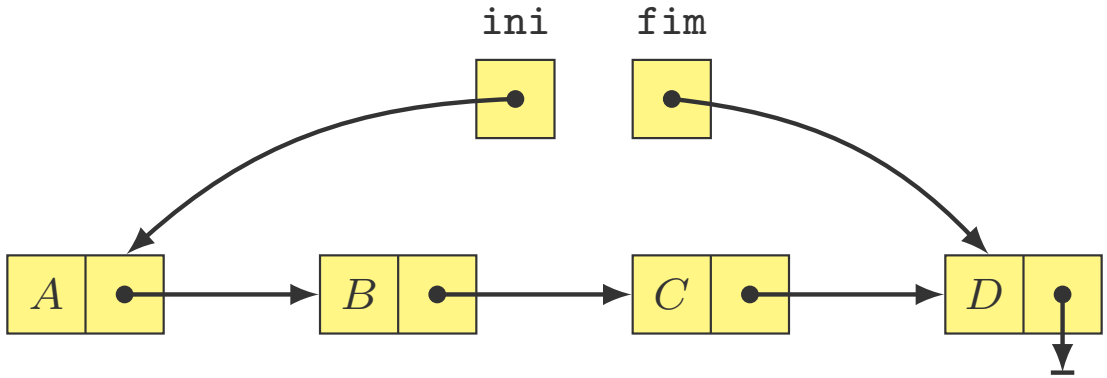
Queue.cpp — Implementação da Fila



Operação `tail()`: Consulta elemento na cauda da lista.

```
1 int Queue::tail() {
```

Queue.cpp — Implementação da Fila



Operação `tail()`: Consulta elemento na cauda da lista.

```
1 int Queue::tail() {
2     if (last != nullptr)
3         return last->key;
4
5     cout << ">> erro: fila vazia" << endl;
6     return INT_MIN;
7 }
```

main.cpp — Programa Principal

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; // Ponteiro para Queue
7
8      fila = new Queue(); // Cria fila do tipo Queue
9
10     for(int i = 1; i <=9; i++)
11         fila->enqueue(i);  // enfileira alguns elementos
12
13     while( !fila->empty() )
14         cout << fila->dequeue() << endl; // desenfileira
15
16     delete fila; // libera memoria alocada para a fila
17
18     return 0;
19 }
```

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

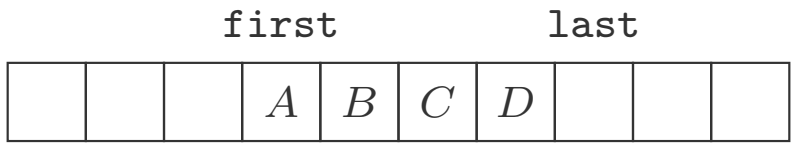
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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 $O(n)$...

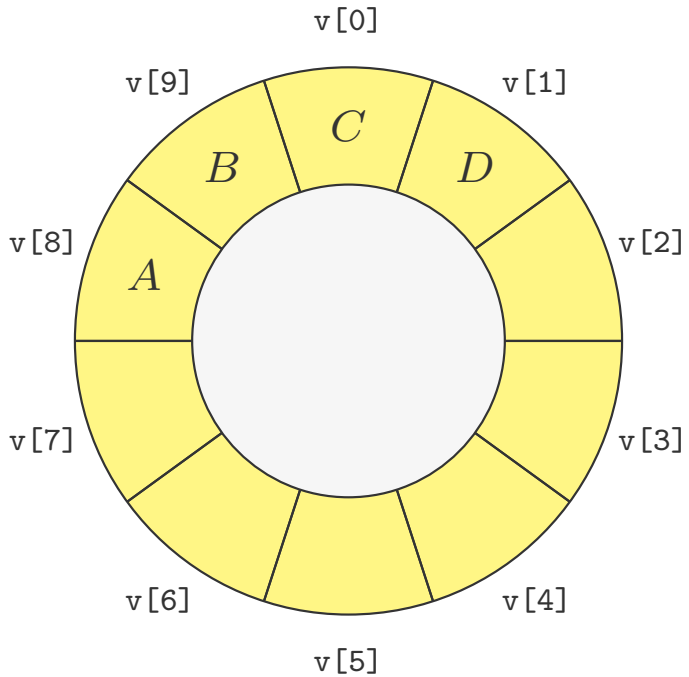
Fila: implementação com vetor (fila circular)

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

Fila: implementação com vetor (fila circular)

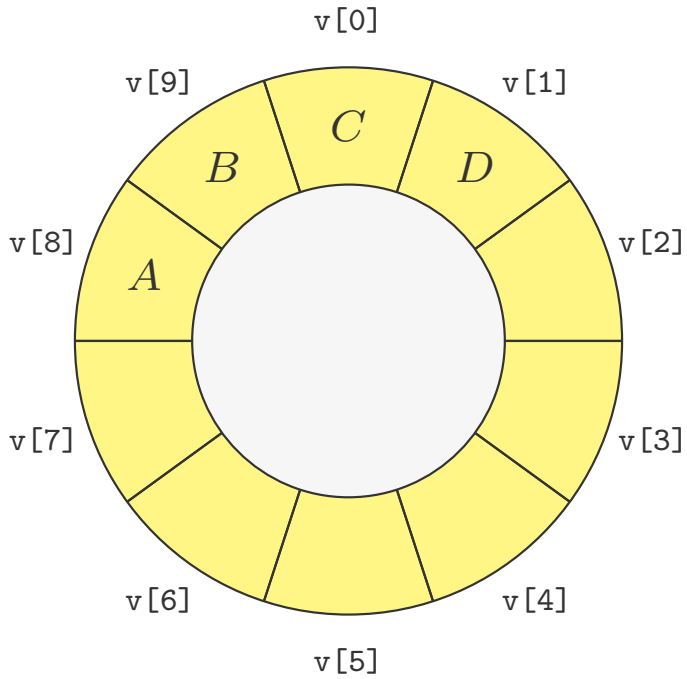
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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  #ifndef QUEUE_ARRAY
2  #define QUEUE_ARRAY
3  #include <iostream>
4
5  class Queue {
6      private:
7          int *array; // ponteiro para vetor de inteiros
8          int size; // Qtd de elementos na Fila
9          int capacity; // Capacidade total da Fila
10         int first; // Indice do primeiro elemento
11     public:
12         Queue(int total); // Construtor
13         ~Queue(); // Destrutor
14         void enqueue(int key); // Adiciona na fila
15         int dequeue(); // Remove da fila
16         bool empty(); // A fila esta vazia?
17         bool full(); // A fila esta cheia?
18         int head(); // Valor do elemento na cabeca
19         int tail(); // valor do elemento na cauda
20 };
21
22 #endif
```

Queue.cpp

Construtor:

```
1  // Construtor
2  Queue::Queue(int total) {
3      array = new (std::nothrow) int[total];
4      if (array == nullptr) {
5          cout << ">> erro: memoria insuficiente." << endl;
6          exit(1);
7      }
8      capacity = total; // capacidade total
9      size = 0; // fila vazia
10     first = 0; // posicao inicial
11 }
```

Queue.cpp

Construtor:

```
1  // Construtor
2  Queue::Queue(int total) {
3      array = new (std::nothrow) int[total];
4      if (array == nullptr) {
5          cout << ">> erro: memoria insuficiente." << endl;
6          exit(1);
7      }
8      capacity = total; // capacidade total
9      size = 0; // fila vazia
10     first = 0; // posicao inicial
11 }
```

Destrutor:

```
1  // Destrutor
2  Queue::~Queue() {
3      if(array != nullptr) delete[] array;
4  }
```

Queue.cpp

Testando se a fila está vazia ou cheia:

Queue.cpp

Testando se a fila está vazia ou cheia:

```
1 bool Queue::empty() {
2     return (size == 0);
3 }
```

Queue.cpp

Testando se a fila está vazia ou cheia:

```
1 bool Queue::empty() {
2     return (size == 0);
3 }
4
5 bool Queue::full() {
6     return (size == capacity);
7 }
```

Queue.cpp

Inserindo na fila:

Queue.cpp

Inserindo na fila:

```
1  void Queue::enqueue(int key) {
2      // fila cheia: capacidade esgotada
3      if (size == capacity) {
4          cerr << ">> Capacidade estorou." << endl;
5          return;
6      }
7      // insere elemento na proxima posicao livre
8      int fim = (first + size) % capacity;
9      array[fim] = key;
10     size++;
11 }
```

Queue.cpp

Removendo da Fila:

Queue.cpp

Removendo da Fila:

```
1  int Queue::dequeue() {
2      if (empty()) {
3          cerr << "fila vazia" << endl;
4          return INT_MIN;
5      }
6      // retira elemento do inicio
7      int value = array[first];
8      first = (first + 1) % capacity;
9      size--;
10     return value;
11 }
```

Queue.cpp

Acessando cabeça e cauda da fila:

```
1 int Queue::head() {
2     if (!empty())
3         return array[first];
4
5     cerr << "erro: fila vazia" << endl;
6     return INT_MIN;
7 }
```

Queue.cpp

Acessando cabeça e cauda da fila:

```
1  int Queue::head() {
2      if (!empty())
3          return array[first];
4
5      cerr << "erro: fila vazia" << endl;
6      return INT_MIN;
7  }
8
9  int Queue::tail() {
10     if (!empty())
11         return array[(first+size-1) % capacity];
12
13     cerr << "erro: fila vazia" << endl;
14     return INT_MIN;
15 }
```

main.cpp — Programa principal

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  #include <iostream>
2  #include "Queue.h"
3  using namespace std;
4  const int MAX = 30;
5
6  int main() {
7      Queue *fila;
8
9      fila = new Queue(MAX); // Criacao de fila
10
11     for (int i = 1; i <= 35; i++)
12         if(!fila->full())
13             fila->enqueue(i);  // enfileirando
14
15     while (!fila->empty()) {
16         cout << "Removeu: " << fila->dequeue() << endl;
17     }
18
19     delete fila; // liberando memoria alocada
20
21     return 0;
22 }
```

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`.

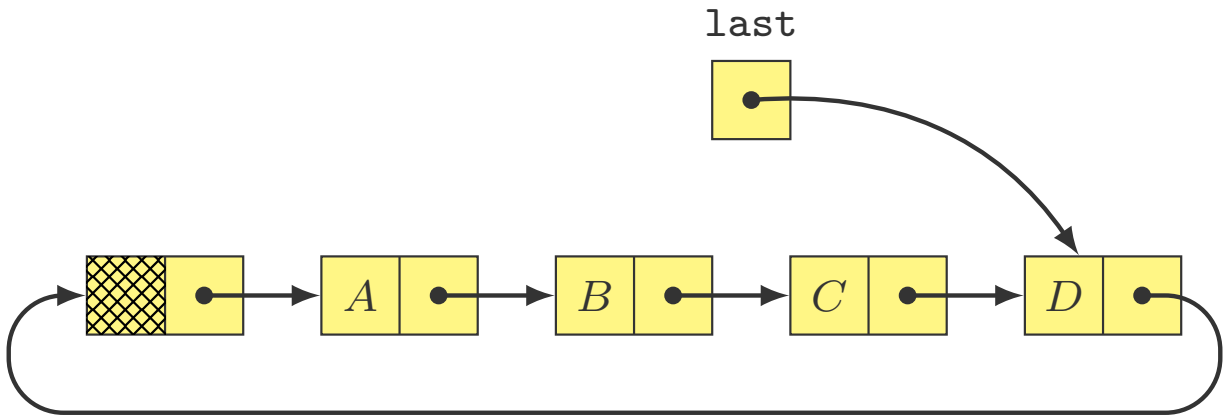
- Note que, ao final dessa função, as filas `f1` e `f2` vão estar vazias, e a fila `f_res` vai conter todos os valores originalmente em `f1` e `f2` (inicialmente `f_res` pode ou não estar vazia).

- 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

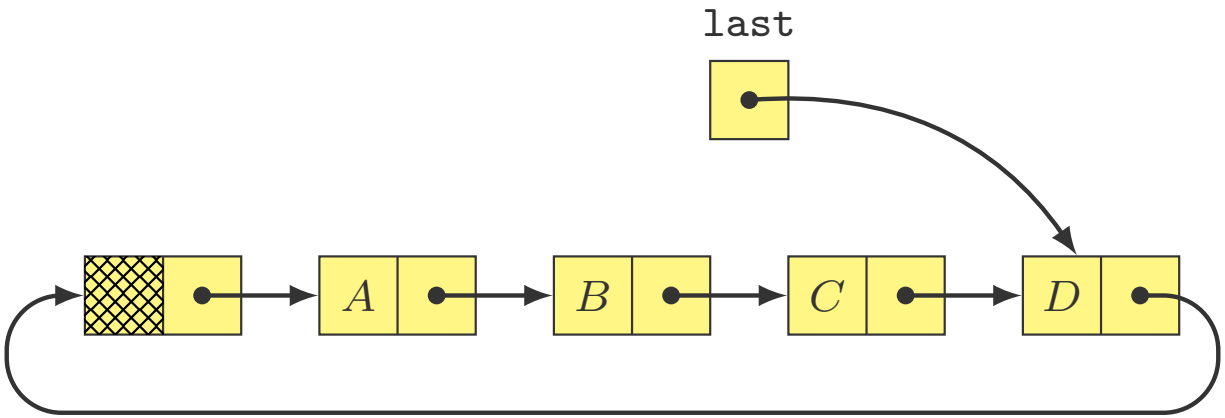
Exercício: implemente uma fila em uma lista encadeada circular com nó cabeça.



Exercício 2 — Implementação Alternativa (Filas)

Universidade Federal do Ceará
Campus Quixadá

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



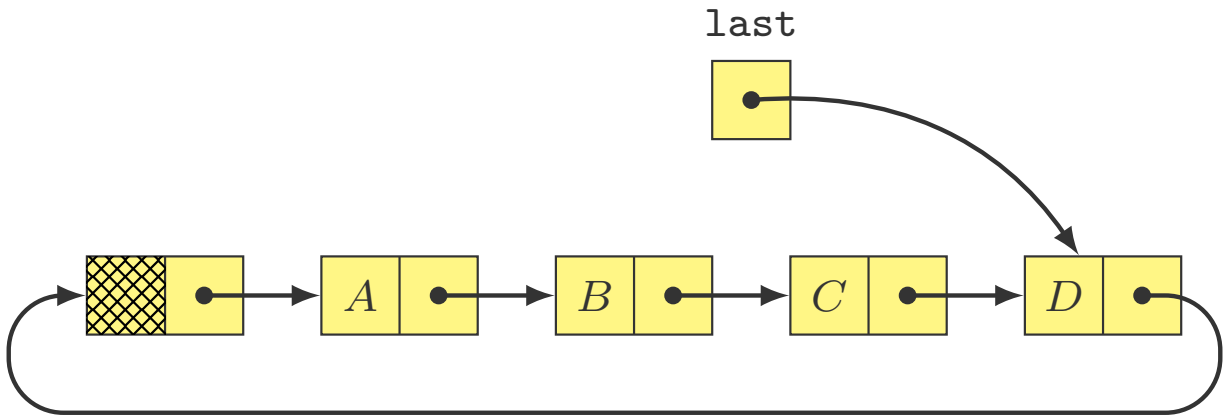
Enfileira:

- 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