Lista Simplesmente Encadeada Circular

Estrutura de Dados — QXD0010

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

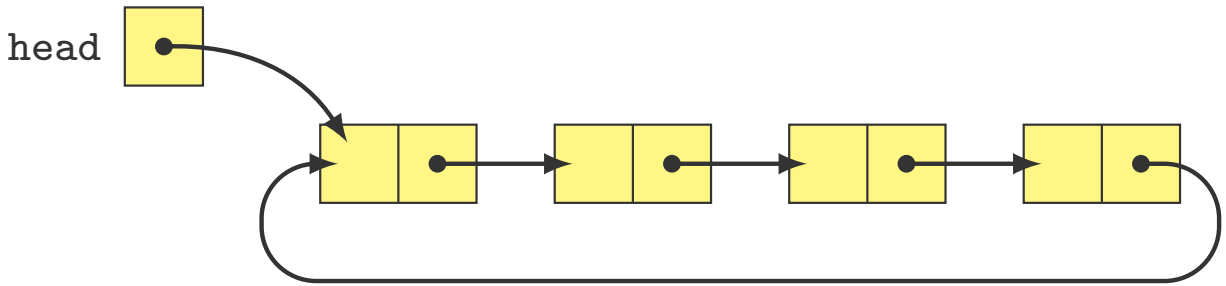
Universidade Federal do Ceará
Campus Quixadá

2º semestre/2022

Introdução

Lista simplesmente encadeada circular

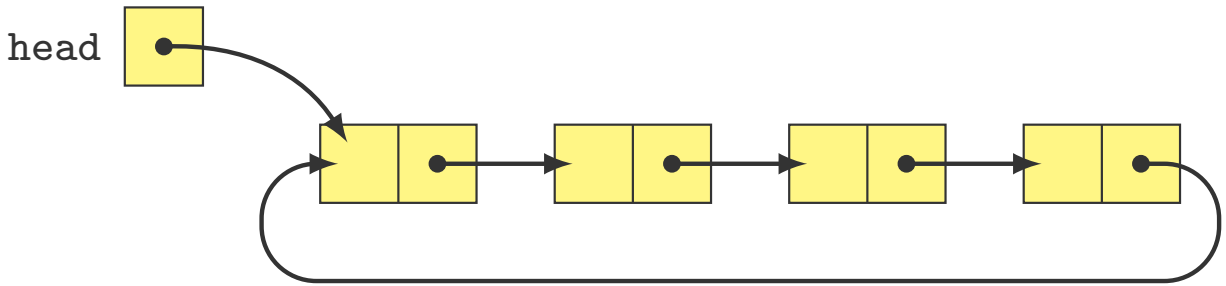
Lista circular (sem nó cabeça):



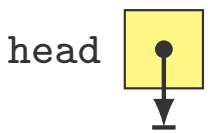
Lista simplesmente encadeada circular

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lista circular (sem nó cabeça):



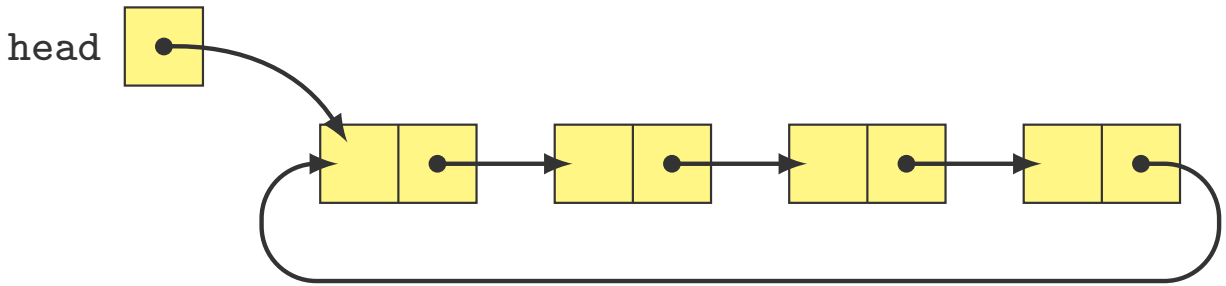
Lista circular vazia: ponteiro `head` é nulo.



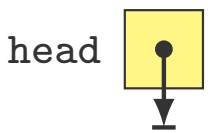
Lista simplesmente encadeada circular

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lista circular (sem nó cabeça):



Lista circular vazia: ponteiro `head` é nulo.

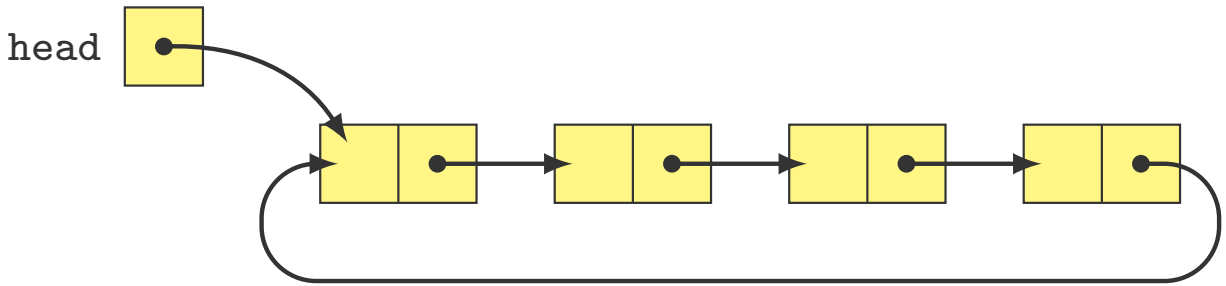


Exemplo de aplicações:

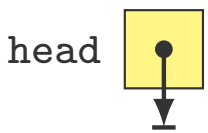
Lista simplesmente encadeada circular

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lista circular (sem nó cabeça):



Lista circular vazia: ponteiro `head` é nulo.



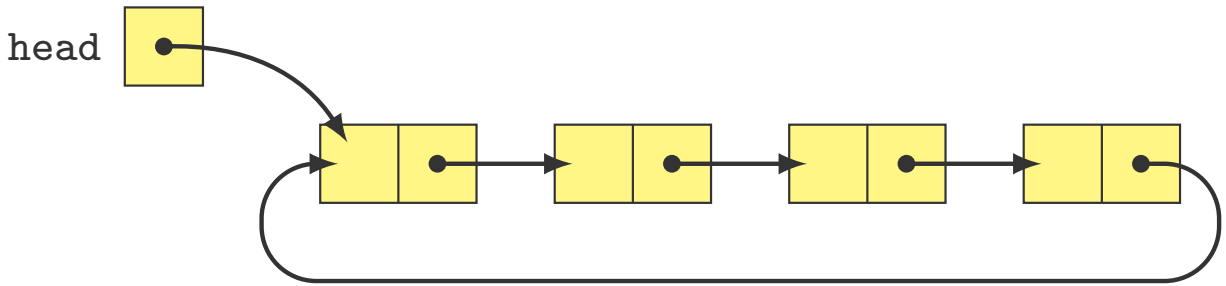
Exemplo de aplicações:

- Execução de processos no sistema operacional

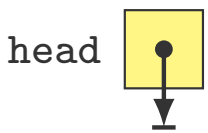
Lista simplesmente encadeada circular

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lista circular (sem nó cabeça):



Lista circular vazia: ponteiro `head` é nulo.



Exemplo de aplicações:

- Execução de processos no sistema operacional
- Controlar de quem é a vez em um jogo de tabuleiro

Implementação em C++

Node.h


```
1  #ifndef NODE_H
2  #define NODE_H
3
4  using Item = double;
5
6  struct Node {
7      Item value;
8      Node* next;
9
10     Node(const Item& val, Node *nextPtr) {
11         value = val;
12         next = nextPtr;
13     }
14 };
15
16 #endif
```

List.h — Arquivo de Cabeçalho


```
1  #ifndef CLIST_H
2  #define CLIST_H
3  #include <string>
4  #include "Node.h"
5
6  class CircularList {
7  private:
8      Node *m_head {nullptr};
9      int m_size {0};
10 public:
11     CircularList() = default;
12     bool empty() const { return m_head == nullptr; }
13     int size() const { return m_size; }
14     void push_back(const Item& val);
15     void pop_back();
16     Item& operator[](int index);
17     const Item& operator[](int index) const;
18     std::string toString() const;
19     void clear();
20     ~CircularList();
21 };
22
23 #endif
```

List.cpp — push_back

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1  #include <sstream>
2  #include <string>
3  #include "Node.h"
4  #include "CircularList.h"
5
6  void CircularList::push_back(const Item& val) {
7      if(m_head == nullptr) { // lista vazia
8          m_head = new Node(val, nullptr);
9          m_head->next = m_head;
10     }
11     else { // lista nao vazia
12         Node *atual = m_head;
13         while(atual->next != m_head)
14             atual = atual->next;
15         atual->next = new Node(val, m_head);
16     }
17     m_size++;
18 }
```

List.cpp — pop_back


```
19 void CircularList::pop_back() {
20     if(m_head == nullptr)   // lista vazia
21         return;
22     if(m_head->next == m_head) { // 1 elemento
23         delete m_head;
24         m_head = nullptr;
25     }
26     else { // 2 ou mais elementos
27         Node *atual = m_head;
28         while(atual->next->next != m_head)
29             atual = atual->next;
30         delete atual->next;
31         atual->next = m_head;
32     }
33     m_size--;
34 }
```

List.cpp — pop_back


```
35 void CircularList::pop_back() {
36     if(m_head == nullptr)   // lista vazia
37         return;
38     if(m_head->next == m_head) { // 1 elemento
39         delete m_head;
40         m_head = nullptr;
41     }
42     else { // 2 ou mais elementos
43         Node *atual = m_head;
44         while(atual->next->next != m_head)
45             atual = atual->next;
46         delete atual->next;
47         atual->next = m_head;
48     }
49     m_size--;
50 }
```

- Qual o tempo de execução dessa função?

List.cpp — toString

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
51 std::string CircularList::toString() const {
52     std::stringstream ss;
53     ss << "[ ";
54     Node *atual = m_head;
55     if(atual != nullptr) {
56         ss << atual->value << " ";
57         while(atual->next != m_head) {
58             atual = atual->next;
59             ss << atual->value << " ";
60         }
61     }
62     ss << "]";
63     return ss.str();
64 }
```

List.cpp — toString


```
65 std::string CircularList::toString() const {
66     std::stringstream ss;
67     ss << "[ ";
68     Node *atual = m_head;
69     if(atual != nullptr) {
70         ss << atual->value << " ";
71         while(atual->next != m_head) {
72             atual = atual->next;
73             ss << atual->value << " ";
74         }
75     }
76     ss << "]";
77     return ss.str();
78 }
```

- Qual o tempo de execução dessa função?

List.cpp — operator[ ]

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
79 Item& CircularList::operator[](int index) {
80     Node *temp = m_head;
81     int count = 0;
82     while(count < index) {
83         count++;
84         temp = temp->next;
85     }
86     return temp->value;
87 }
88
89 const Item& CircularList::operator[](int index) const {
90     Node *temp = m_head;
91     int count = 0;
92     while(count < index) {
93         count++;
94         temp = temp->next;
95     }
96     return temp->value;
97 }
```

List.cpp — Liberando todos os nós

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
 98 void CircularList::clear() {
 99     if(m_head != nullptr) {
100         Node *aux = m_head->next;
101         while(aux != m_head) {
102             Node *t = aux;
103             aux = aux->next;
104             delete t;
105         }
106         delete m_head;
107         m_head = nullptr;
108         m_size = 0;
109     }
110 }
```

List.cpp — Liberando todos os nós

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
114 void CircularList::clear() {
115     if(m_head != nullptr) {
116         Node *aux = m_head->next;
117         while(aux != m_head) {
118             Node *t = aux;
119             aux = aux->next;
120             delete t;
121         }
122         delete m_head;
123         m_head = nullptr;
124         m_size = 0;
125     }
126 }

127 CircularList::~CircularList() {
128     clear();
129 }
```

main.cpp — Programa Cliente

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

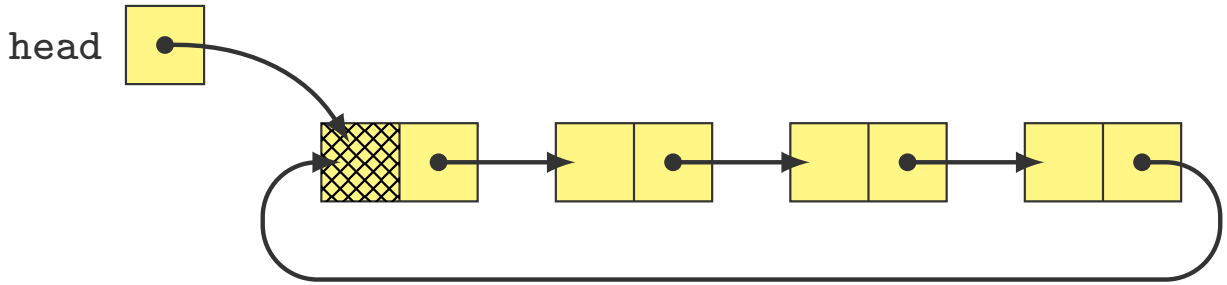
```
1  #include <iostream>
2  #include "CircularList.h"
3  using namespace std;
4
5  void print(const CircularList& lst) {
6      for(int i = 0; i < lst.size(); ++i)
7          cout << lst[i] << " ";
8      cout << endl;
9  }
10
11 int main() {
12     CircularList lst;
13
14     for(int i = 1; i <= 9; ++i)
15         lst.push_back(i * 0.5);
16
17     cout << lst.toString() << endl;
18 }
```

Variações

Variações — Listas circulares com nó sentinela

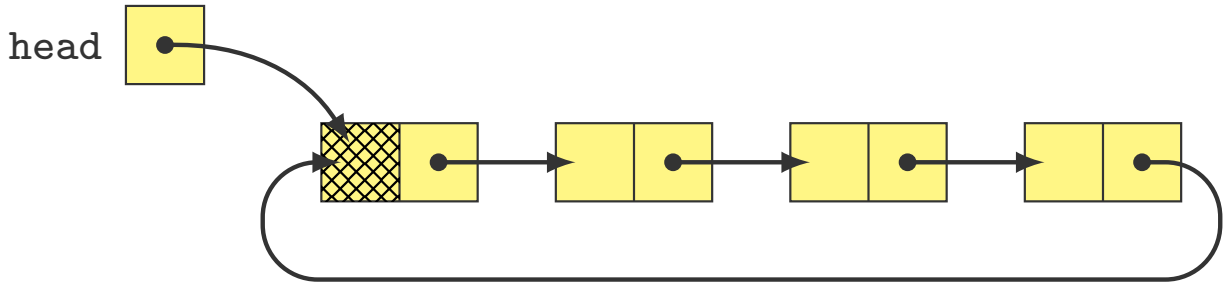
Universidade Federal do Ceará
Campus Quixadá

Lista circular com nó sentinela:

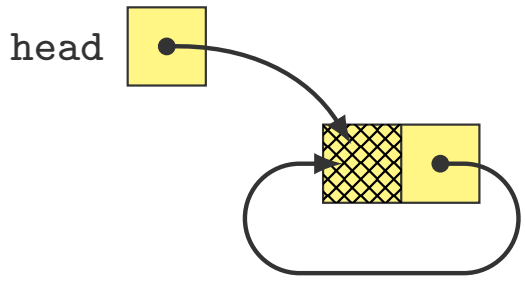


Variações — Listas circulares com nó sentinela

Lista circular com nó sentinela:

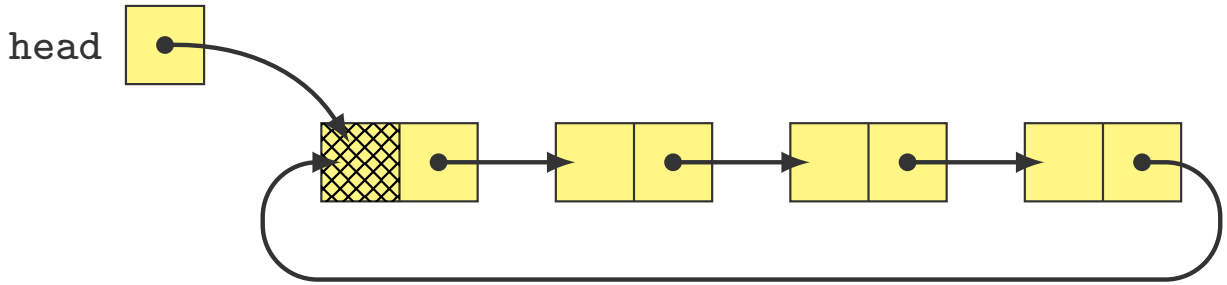


Lista circular vazia com nó sentinela:

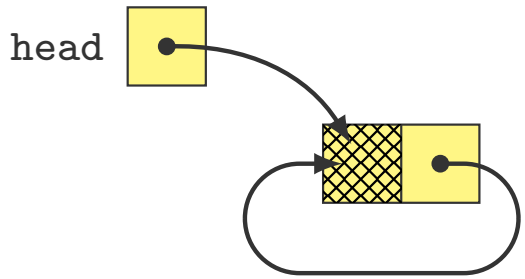


Variações — Listas circulares com nó sentinela

Lista circular com nó sentinela:



Lista circular vazia com nó sentinela:

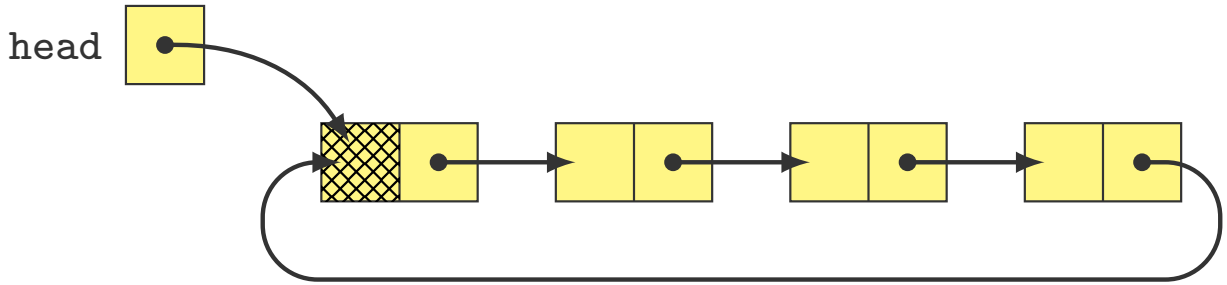


Diferenças para a versão sem nó sentinela:

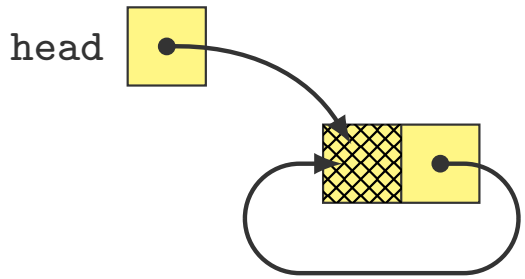
Variações — Listas circulares com nó sentinela

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lista circular com nó sentinela:



Lista circular vazia com nó sentinela:



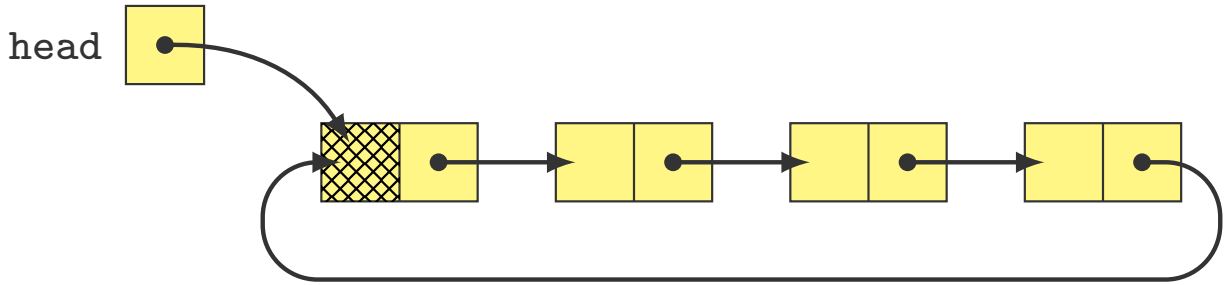
Diferenças para a versão sem nó sentinela:

- ponteiro `head` sempre aponta para nó sentinela

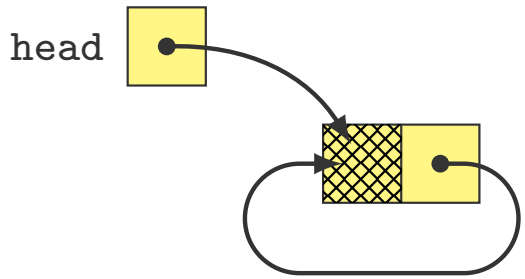
Variações — Listas circulares com nó sentinela

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lista circular com nó sentinela:



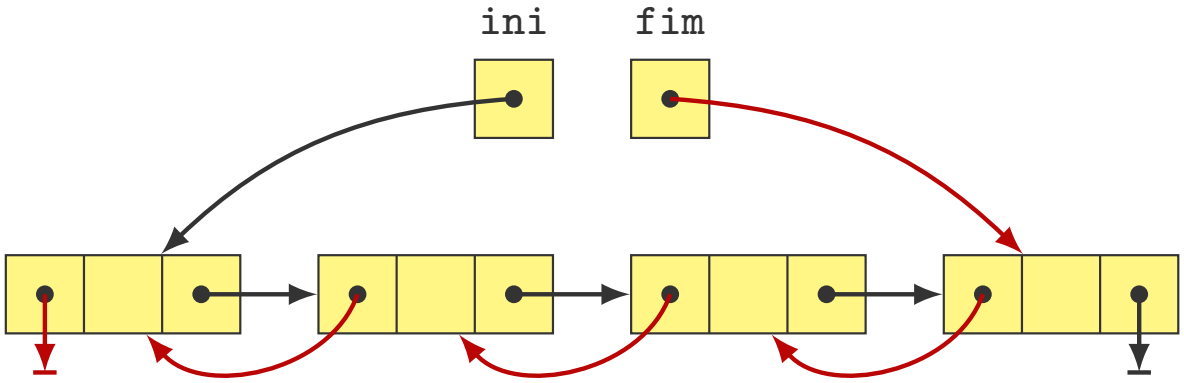
Lista circular vazia com nó sentinela:



Diferenças para a versão sem nó sentinela:

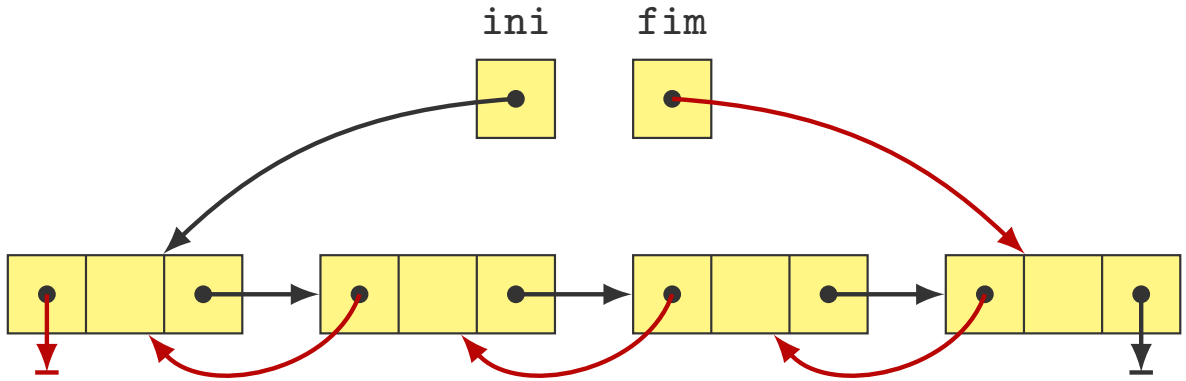
- ponteiro `head` sempre aponta para nó sentinela
- código de inserção e de remoção mais simples

Variações - Lista duplamente encadeada



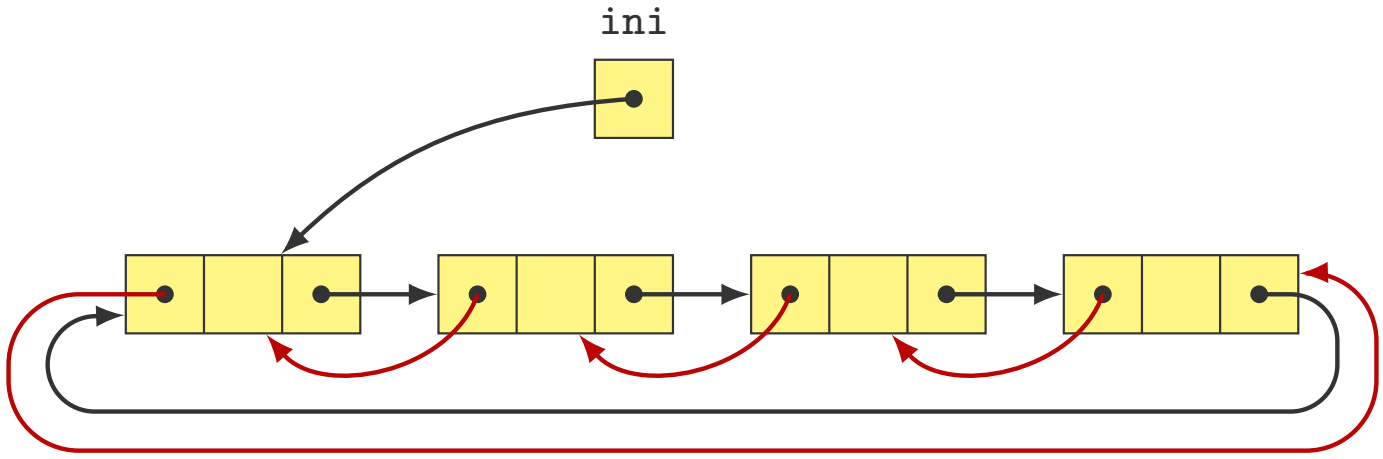
Variações - Lista duplamente encadeada

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



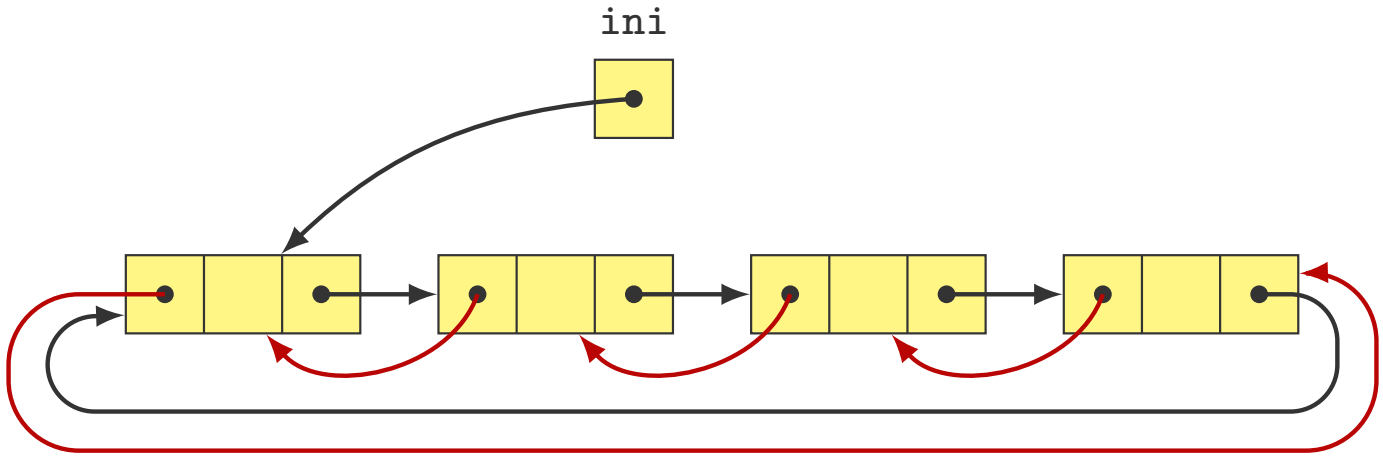
- Cada nó tem um ponteiro para o próximo nó e para o nó anterior.
- Se tivermos um ponteiro para o último elemento da lista, podemos percorrer a lista em ordem reversa.

Variações - Lista circular duplamente encadeada



Variações - Lista circular duplamente encadeada

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



Permite inserção e remoção em $O(1)$

Podemos ter uma lista dupla circular com cabeça também...

Exercícios

Exercícios

- Implemente uma lista duplamente encadeada com as operações:
  - inserir nó
  - remover nó
  - saber se há nó com dado valor
  - tamanho da lista
  - concatenar duas listas
  - imprimir lista de frente para trás ou reversamente

- Implemente uma lista circular duplamente encadeada com as operações:
  - inserir nó
  - remover nó
  - saber se há nó com dado valor
  - tamanho da lista
  - concatenar duas listas
  - imprimir lista de frente para trás ou reversamente

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

FIM