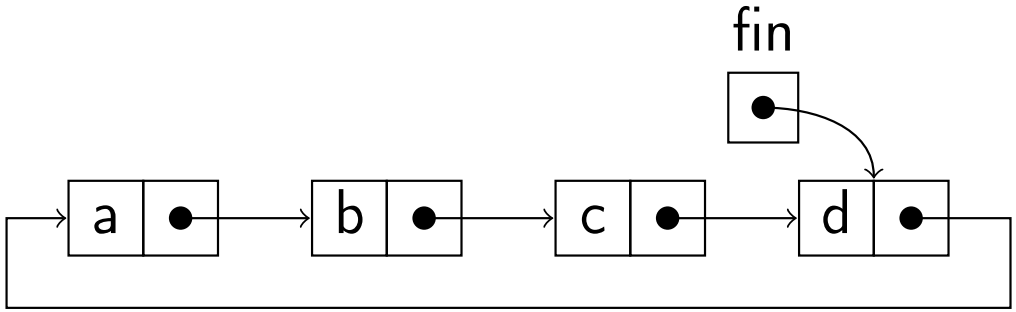
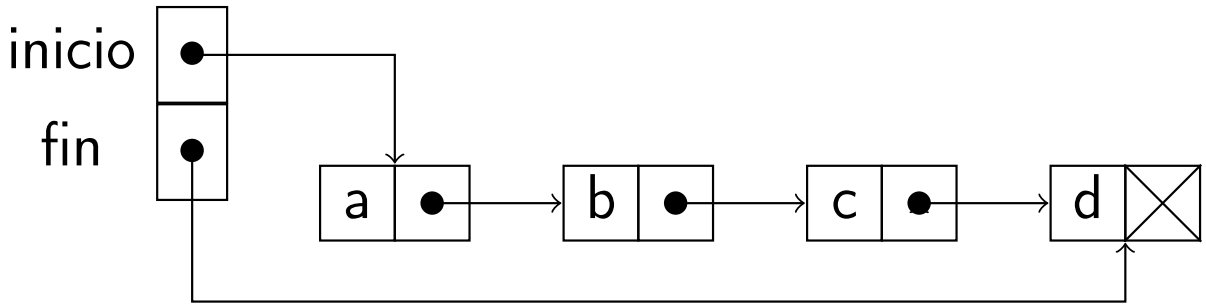
Implementación mediante una estructura enlazada

Estructura dinámica

El tamaño de la estructura de datos varía en tiempo de ejecución con el tamaño de la cola. A cambio se ocupa espacio adicional con los enlaces.



(a) Estructura enlazada circular



(b) Dos punteros a los extremos

Implementación mediante una estructura enlazada

```
1  #ifndef COLA_ENLA_H
2  #define COLA_ENLA_H
3  #include <cassert>

5  template <typename T> class Cola {
6  public:
7     Cola(); // constructor
8     Cola(const Cola<T>& C); // ctor. de copia
9     Cola<T>& operator =(const Cola<T>& C); // asignación de colas
10    bool vacia() const;
11    const T& frente() const;
12    void pop();
13    void push(const T& x);
14    ~Cola(); // destructor
```

Implementación mediante una estructura enlazada

```
15 private:
16     struct nodo {
17         T elto;
18         nodo* sig;
19         nodo(const T& e, nodo* p = 0): elto(e), sig(p) {}
20     };

22     nodo *inicio, *fin; // extremos de la cola

24     void copiar(const Cola<T>& C);
25 };
```

Implementación mediante una estructura enlazada

```
26 // Método privado
27 template <typename T>
28 void Cola<T>::copiar(const Cola<T>& C)
29 {
30    if (C.inicio) { // C no está vacía
31        // Copiar el primer elto.
32       inicio = fin = new nodo(C.inicio->elto);
33       // Copiar el resto de elementos hasta el final de la cola.
34       for (nodo *p = C.inicio->sig; p; p = p->sig) {
35          fin->sig = new nodo(p->elto);
36          fin = fin->sig;
37       }
38    }
39 }
```

Implementación mediante una estructura enlazada

```
41 template <typename T>
42 inline Cola<T>::Cola() : inicio(0), fin(0) {}

44 template <typename T>
45 inline Cola<T>::Cola(const Cola<T>& C) : inicio(0), fin(0)
46 {
47    copiar(C);
48 }

50 template <typename T>
51 inline Cola<T>& Cola<T>::operator =(const Cola<T>& C)
52 {
53    if (this != &C) { // evitar autoasignación
54       this->~Cola(); // vaciar la cola actual
55       copiar(C);
56    }
57    return *this;
58 }
```

Implementación mediante una estructura enlazada

```
60 template <typename T>
61 inline bool Cola<T>::vacia() const
62 {
63    return (inicio == 0);
64 }

66 template <typename T>
67 inline const T& Cola<T>::frente() const
68 {
69    assert(!vacia());
70    return inicio->elto;
71 }
```

Implementación mediante una estructura enlazada

```
73 template <typename T>
74 inline void Cola<T>::pop()
75 {
76    assert(!vacia());
77    nodo* p = inicio;
78    inicio = p->sig;
79    if (!inicio) fin = 0;
80    delete p;
81 }

83 template <typename T>
84 inline void Cola<T>::push(const T& x)
85 {
86    nodo* p = new nodo(x);
87    if (inicio == 0) // cola vacía
88       inicio = fin = p;
89    else
90       fin = fin->sig = p;
91 }
```

Implementación mediante una estructura enlazada

```
93  // Destructor: vacía la cola
94  template <typename T>
95  Cola<T>::~Cola()
96  {
97     nodo* p;
98     while (inicio) {
99        p = inicio->sig;
100       delete inicio;
101       inicio = p;
102    }
103    fin = 0;
104 }

106 #endif // COLA_ENLA_H
```