TEMA 4: TIPO DE DATOS ABSTRACTO *LISTA DOBLEMENTE ENLAZADA*

ALGORITMOS Y ESTRUCTURAS DE DATOS

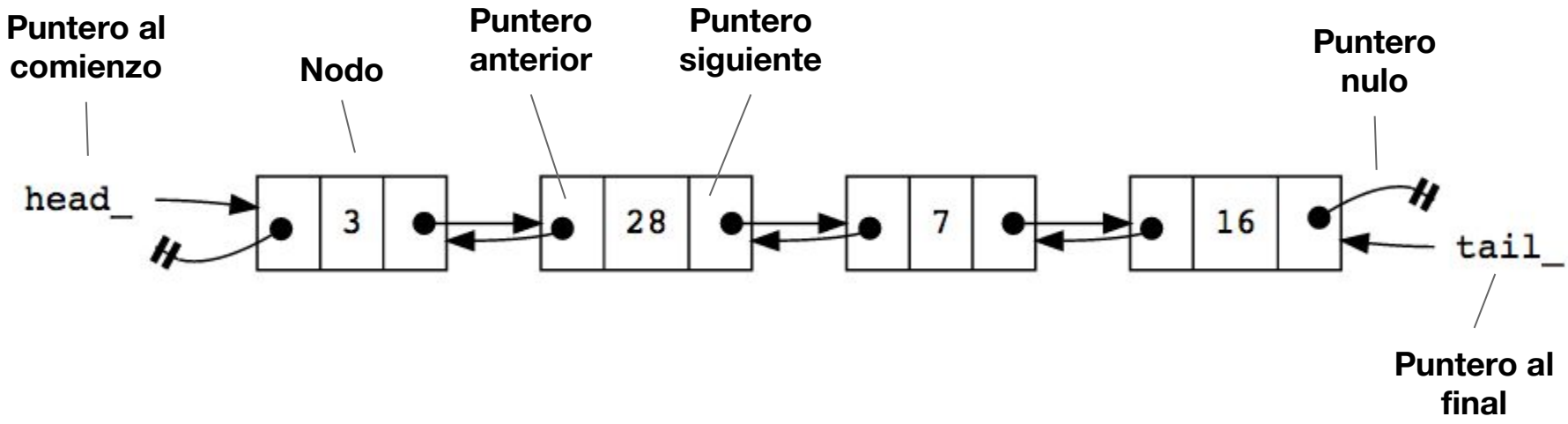
M. Colebrook Santamaría
J. Riera Ledesma

Objetivos

- Especificación formal del TAD lista doblemente enlazada.
- Implementación del TAD lista doblemente enlazada mediante objetos dinámicos.
- Operaciones sobre listas doblemente enlazadas:
  - Inserción
  - Extracción
  - Recorrido
  - Búsqueda
- Implementación de una lista ordenada.
- Implementación de una lista circular.

Especificación formal del TAD lista doblemente enlazada

- Una **lista doblemente enlazada** (*doubly linked list, **dll***) como una **secuencia de nodos**, y en cada nodo se guardan **datos** y dos **punteros**: uno al nodo **anterior** y otro al nodo **posterior**.

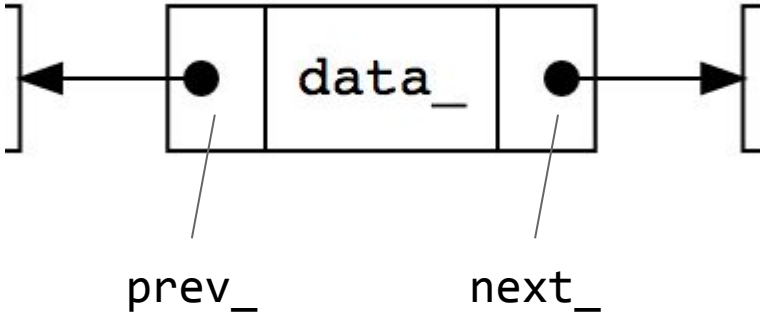


Especificación formal del TAD lista (simplemente) enlazada (4)

- La lista doblemente enlazada tiene las siguientes operaciones (**interfaz pública**):
  - Insertar un nuevo nodo al comienzo/final de la lista.
  - Extraer un nodo del comienzo/final de la lista.
  - Extraer un nodo determinado.
  - Comprobar si la lista está vacía.
- La **implementación** de estas operaciones se desarrollará usando dos clases:
  - Clase nodo `dll_node_t`.
  - Clase lista doblemente enlazada `dll_t`.

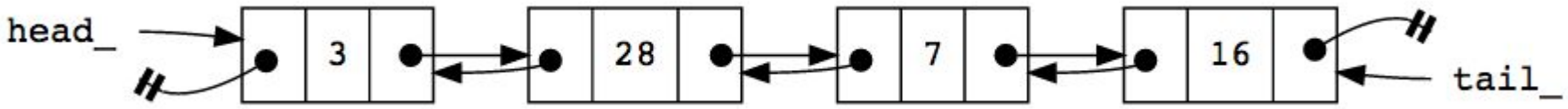
Implementación del TAD lista mediante objetos dinámicos (1)

```
template <class T>
class dll_node_t
{
private:
  dll_node_t<T>* prev_;
  T              data_;
  dll_node_t<T>* next_;

public:
  ...
};
```

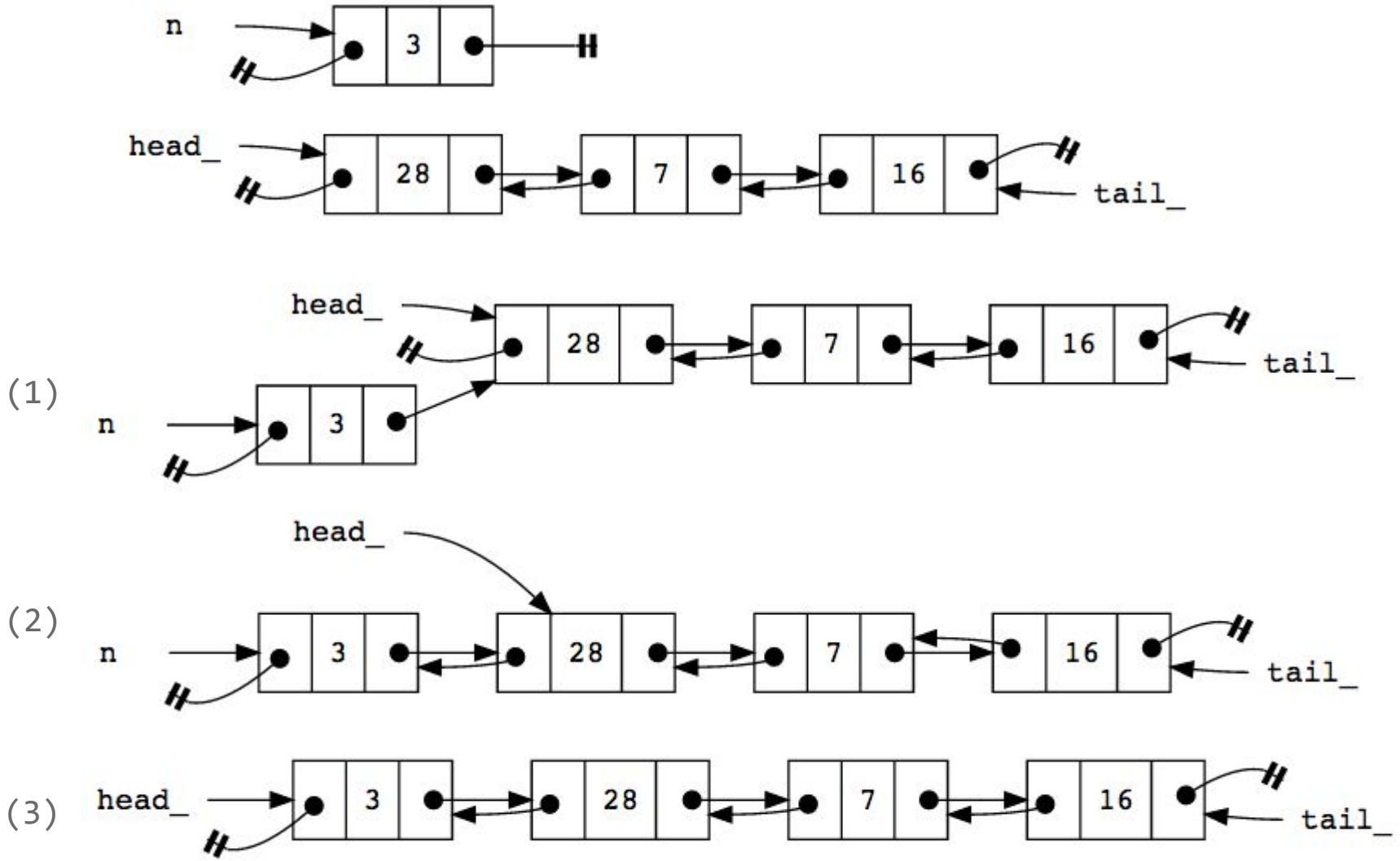


Implementación del TAD lista mediante objetos dinámicos (2)

```
template <class T>
class dll_t
{
private:
  dll_node_t<T>* head_;
  dll_node_t<T>* tail_;
  int sz_;

public:
  ...
};
```



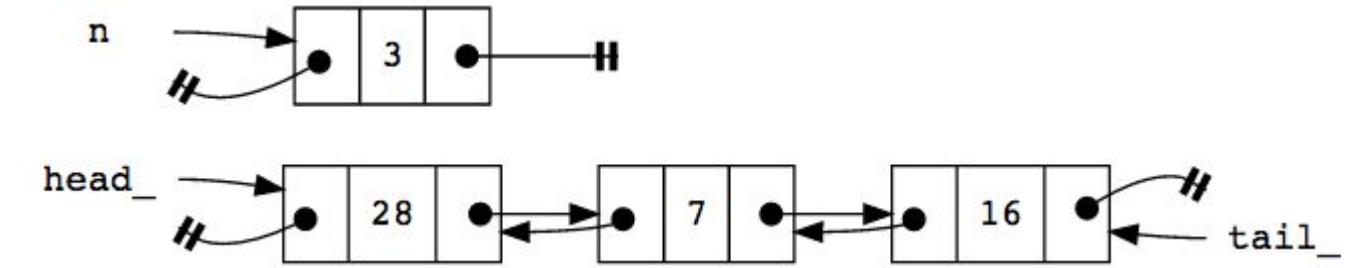
Operaciones: Inserción por el comienzo (1)



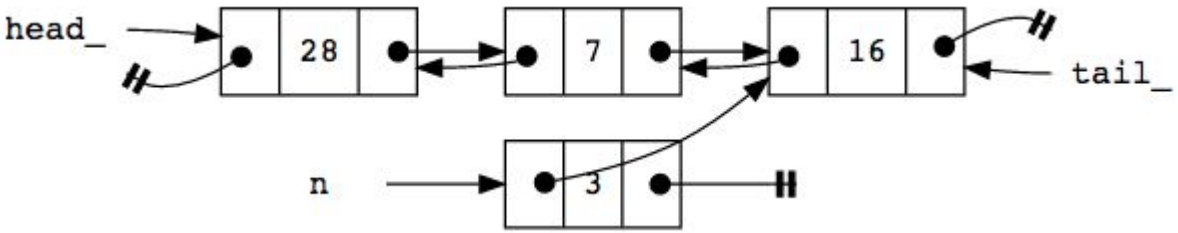
Operaciones: Inserción por el comienzo (2)

```
template <class T>
void dll_t<T>::insert_head(dll_node_t<T>* n)
{
  if (empty()) head_ = tail_ = n;
  else {
    n->set_next(head_);  // (1)
    head_->set_prev(n);  // (2)
    head_ = n;           // (3)
  }
  sz_++;
}
```

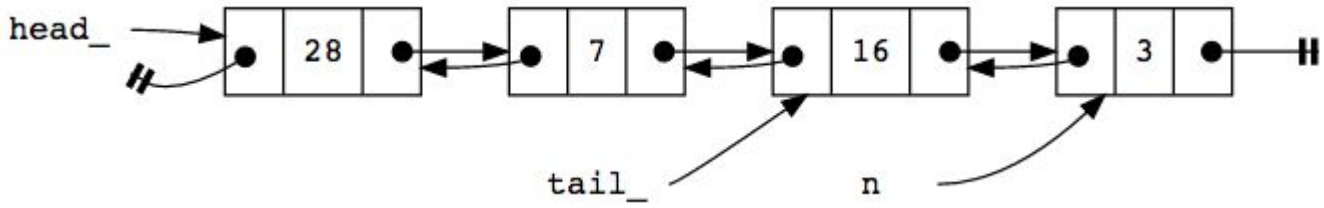
Operaciones: Inserción por el final (1)



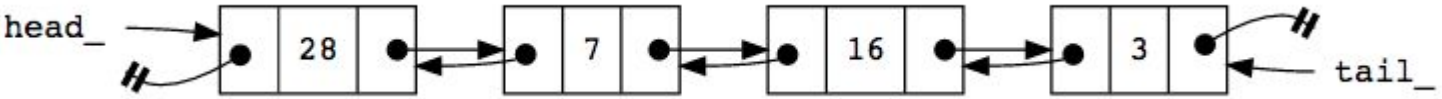
(1)



(2)



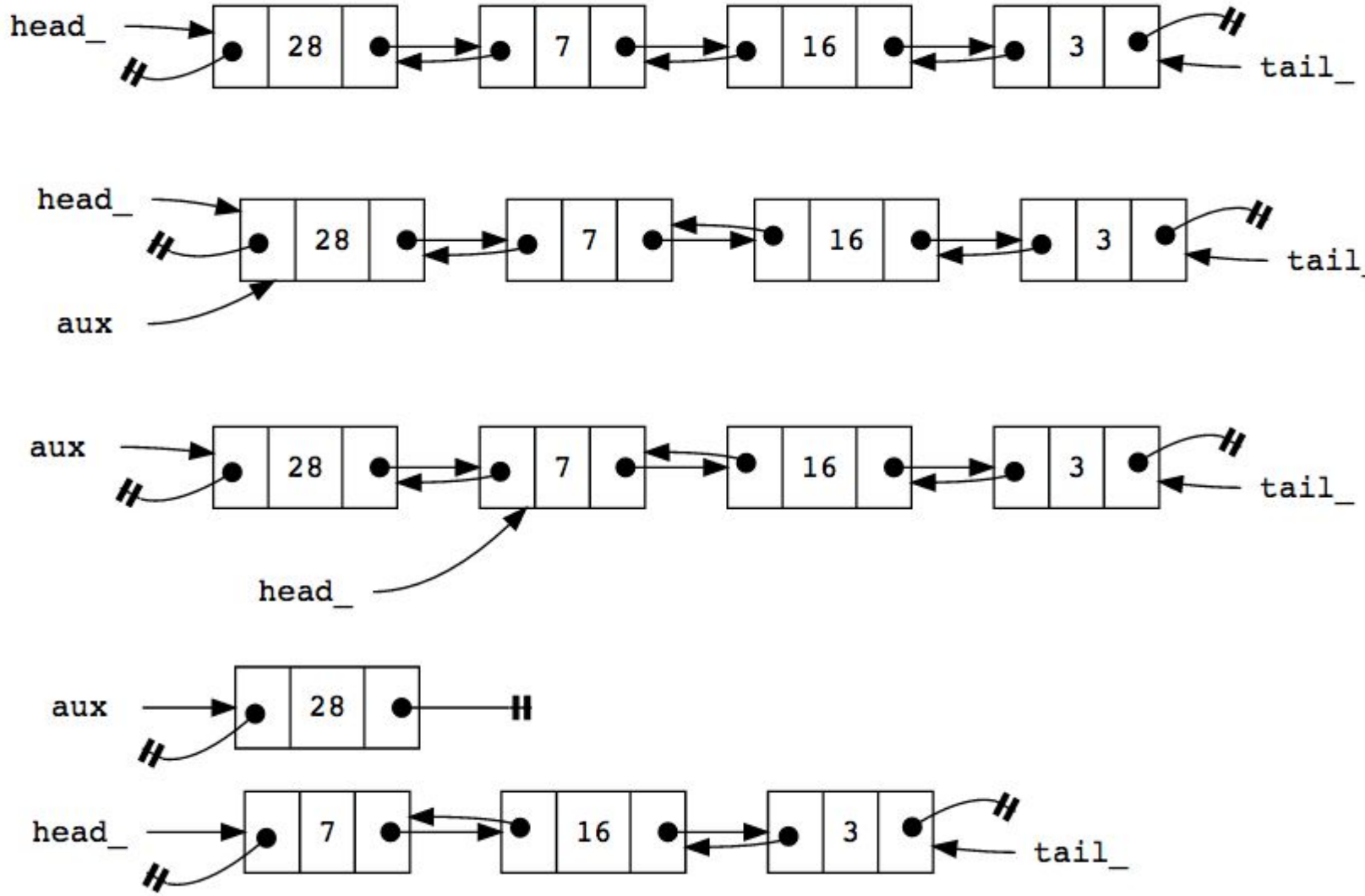
(3)



Operaciones: Inserción por el final (2)

```
template <class T>
void dll_t<T>::insert_tail(dll_node_t<T>* n)
{
  if (empty()) head_ = tail_ = n;
  else {
    n->set_prev(tail_);  // (1)
    tail_->set_next(n);  // (2)
    tail_ = n;           // (3)
  }
  sz_++;
}
```

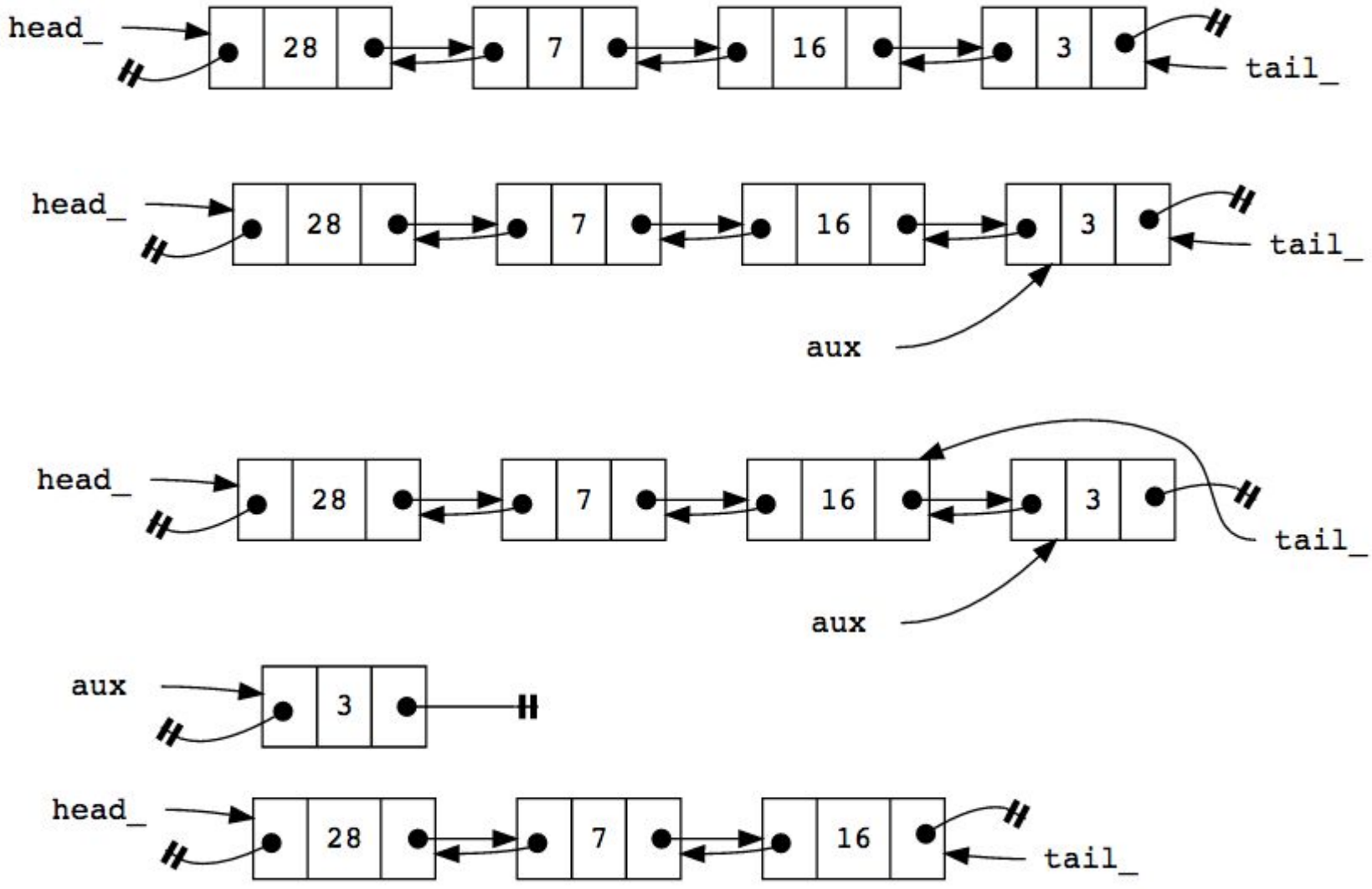
Operaciones: Extracción por el comienzo (1)



Operaciones: Extracción por el comienzo (2)

```
template <class T>
dll_node_t<T>* dll_t<T>::extract_head(void)
{ assert(!empty());
  dll_node_t<T>* aux = head_;  // (1)
  head_ = head_->get_next();   // (2)

  if (head_ != NULL) head_->set_prev(NULL);  // (3)
  else               tail_ = NULL;
  sz--;

  aux->set_next(NULL);
  return aux;
}
```

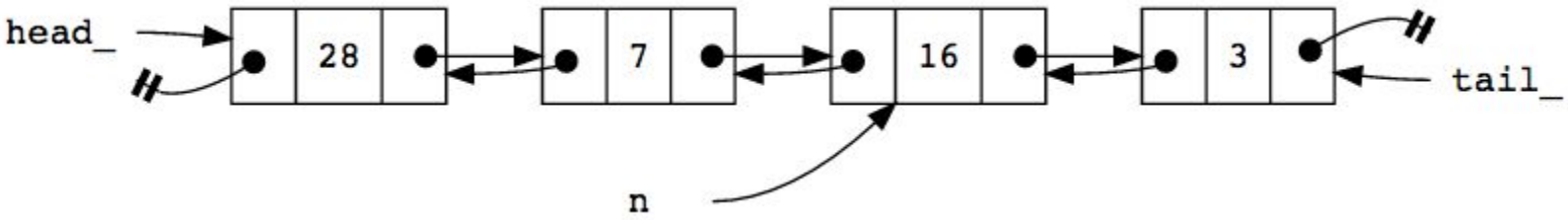
Operaciones: Extracción por el final (1)



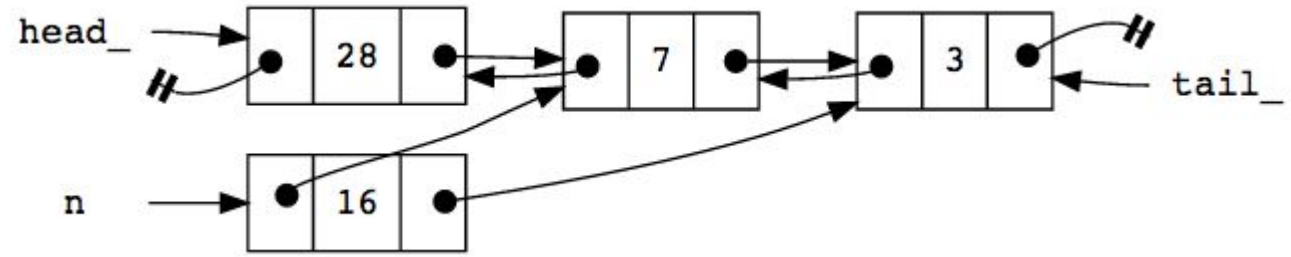
Operaciones: Extracción por el final (2)

```
template <class T>
dll_node_t<T>* dll_t<T>::extract_tail(void)
{ assert(!empty());
  dll_node_t<T>* aux = tail_;  // (1)
  tail_ = tail_->get_prev();   // (2)

  if (tail_ != NULL) tail_->set_next(NULL);  // (3)
  else               head_ = NULL;
  sz_--;

  aux->set_prev(NULL);
  return aux;
}
```

Operaciones: Eliminación de un nodo determinado (1)



(1)



(2)

Operaciones: Eliminación de un nodo determinado (2)

```
template <class T>
void dll_t<T>::remove(dll_node_t<T>* n)
{ assert(n != NULL);
  if (n->get_prev() != NULL) // n != head_
    n->get_prev()->set_next(n->get_next());  // (1)
  else { head_ = n->get_next(); head_->set_prev(NULL); // extract_head()
       }

  if (n->get_next() != NULL) // n != tail_
    n->get_next()->set_prev(n->get_prev());  // (1)
  else { tail_ = n->get_prev(); tail_->set_next(NULL); // extract_tail()
       }
  delete n;  // (2)
  sz--;
}
```

Referencias

- Olsson, M. (2015), “C++ 14 Quick Syntax Reference”, Apress. Disponible en PDF en la BBTK-ULL: absysnetweb.bbtk.ull.es/cgi-bin/abnetopac01?TITN=533049
- Stroustrup, B. (2002), “El Lenguaje de Programación C++”, Addison Wesley.
- C++ Syntax Highlighting (código en colores): tohtml.com/cpp
- Gráficos de las listas: www.webgraphviz.com