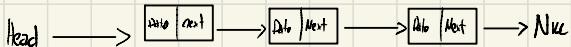
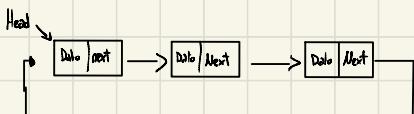


Liste Concatenate

-Lista singolarmente concatenata:



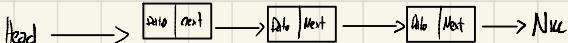
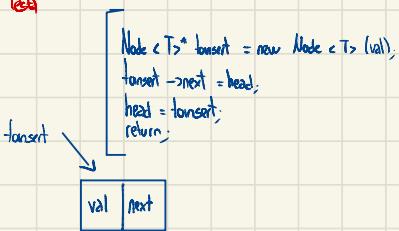
Lista circolare:



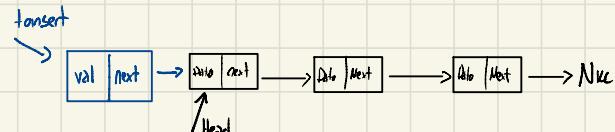
Lista Doppialmente concatenata:



Istrumento in Testa

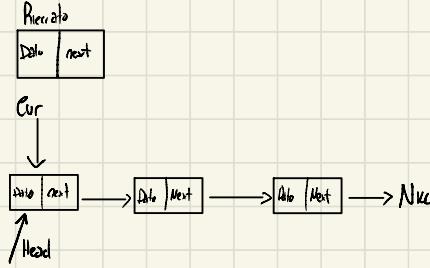


`tinsert->next = head // il next del nuovo nodo punta a head (che sarebbe il puntatore al primo nodo)`



Ricerca:

```
Node<T>* search (T val) {
    if (isEmpty())
        throw out_of_range ("... list is Empty")
    if (head->val == val) // Se la testa punta al valore cercato
        return head; // Ritorniamo Testa
    Node<T>* cur = head; // Scegliamo un puntatore cur che punta alla testa
    while (cur->next && cur->val != val) // Finché cur punta al next (mette spazio al dato del Nodo Successore e il valore trovato non è quello cercato)
        cur = cur->next; // cur punta al prossimo Nodo
    if (cur->val != val) // se il valore non è in nessun Nodo cur è "No Node";
    {
        cerr << "Element with key " << val << " not found" << endl;
        return nullptr;
    }
    return cur;
}
```



Inserimento in Coda:

```
void insertTail (T val) {
    if (isEmpty())
        insertHead (val);
    else
    {
        Node<T>* lastset = new Node<T>(val);
        Node<T>* cur = head;
        while (cur->next)
            cur = cur->next;
        cur->next = lastset;
        lastset->next = cur;
    }
}
```

Promozione di un Nodo:

Void removeHead () {

if (isEmpty ())
{

err << "Empty List! Operation isn't available" << endl;

return;

}

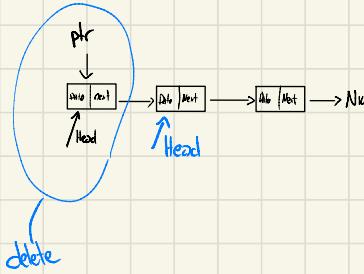
Node < T >* ptr = head; // Puntatore che punta alla Testa.

head = head->next; // head punta a next e lo sposta al nodo successivo

delete ptr;

return;

}



Void RemoveTail () {

if (isEmpty ())
{

err << "Empty List! Operation isn't available" << endl;

return;

}

Node < T >* cur = head;

Node < T >* prev = nullptr;

while (cur->next) // (Fermo finché cur non arriverà all'ultimo nodo)

prev = cur; // prev punta al nodo prima.

cur = cur->next;

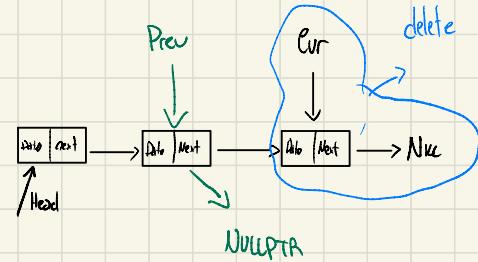
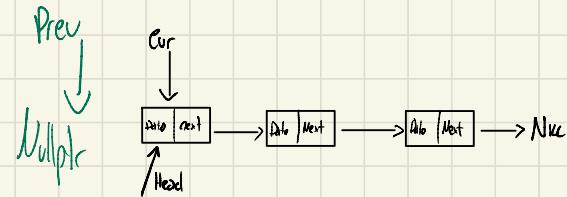
}

prev->next = nullptr; // Il penultimo nodo ora punta Null

delete cur; // Eliminato ultimo nodo in lista

return;

}



Cancellazione Elemento:

```
void RemoveElement(T val)
{
}
```

```
if (IsEmpty())
    return;
```

Node<T>* toremove = search(val); // Punto il Nodo da Cancellare
 ↗ crea il
Valore da Cancellare

```
f(!toremove)
```

return; // Se non viene trovato esce dalla funzione!

```
if (toremove->val == head->val) // Se l'elemento è in testa  
richiamo
```

RemoveHead ed esce

```
removeHead();
```

```
return;
```

```
}
```

Node<T>* cur = head; Cur punta a testa

Node<T>* prev = nullptr;

```
while (cur->next != nullptr && cur->val != toremove->val)
```

```
{
```

Prev = cur; // Il precedente sarà in posizione
del 'corrente'

Cur = cur->next; // Corrente si sposta al Nodo Suce...

```
}
```

prev->next = cur->next; // Il puntatore 'prev' punta al 'next'

delete cur; // che è assegnato al puntatore 'next'
di 'cur' perché puntando a 'next'
punterà al nodo successivo di 'cur'

→ Eliminiamo
il nodo cercato.

Rimozione di un Nodo nella parte Centrale:

Da cancellare 

prev -> next

Cur

[12 | next] —> [17 | Next] —> [86 | Next] —> [53 | Next] —> NULL

prev cur

↓

[12 | next] —> [17 | Next] —> [86 | Next] —> [53 | Next] —> NULL

prev

Cur

[12 | next] —> [17 | Next] —> [86 | Next] —> [53 | Next] —> NULL

Venne ricreato il filo fino a quando non lo trovammo.

Una volta trovato?

