

## **BÁO CÁO ĐỒ ÁN LINKED LIST**

### **1. Hàm createNode(int data)**

**Mục tiêu:** Tạo một node mới chứa dữ liệu data.

**Cách giải:**

Cấp phát vùng nhớ mới bằng new.

Gán key = data và p\_next = NULL để tạo node cuối.

Trả về con trỏ đến node vừa tạo.

### **2. Hàm createList(NODE \*p\_node)**

**Mục tiêu:** Khởi tạo danh sách liên kết với một node đầu tiên.

**Cách giải:**

Tạo danh sách mới bằng new List().

Nếu p\_node == NULL, danh sách rỗng.

Ngược lại, p\_head = p\_tail = p\_node, và p\_node->p\_next = NULL.

### **3. Hàm addHead(List\*& L, int data)**

**Mục tiêu:** Thêm node mới vào đầu danh sách.

**Cách giải:**

Tạo node mới.

Nếu danh sách rỗng, gán p\_head = p\_tail = node.

Nếu không, cho node trỏ đến p\_head và cập nhật lại p\_head.

### **4. Hàm addTail(List\*& L, int data)**

**Mục tiêu:** Thêm node vào cuối danh sách.

**Cách giải:**

Tạo node mới.

Nếu danh sách rỗng, gán p\_head = p\_tail = node.

Nếu không, p\_tail->p\_next = node, rồi cập nhật p\_tail.

### **5. Hàm removeHead(List\*& L)**

**Mục tiêu:** Xóa node đầu danh sách.

**Cách giải:**

Nếu rỗng thì return false.

Lưu node đầu, cập nhật p\_head = p\_head->p\_next, xóa node cũ.

Nếu danh sách còn 1 node thì p\_tail = NULL.

#### 6. Hàm removeTail(List\*& L)

**Mục tiêu:** Xóa node cuối cùng.

**Cách giải:**

Nếu danh sách rỗng hoặc có 1 node, xóa như removeHead.

Duyệt từ đầu đến node kế cuối.

Cập nhật p\_tail, xóa node cuối.

#### 7. Hàm removeAll(List\*& L)

**Mục tiêu:** Xóa toàn bộ danh sách.

**Cách giải:**

Duyệt từng node, xóa bằng delete.

Sau cùng, gán p\_head = p\_tail = NULL.

#### 8. Hàm removeBefore(List\*& L, int val)

**Mục tiêu:** Xóa node trước node có giá trị val.

**Cách giải:**

Duyệt theo 3 con trỏ prev, cur, next.

Nếu next->p\_next->key == val, thì delete(next).

#### 9. Hàm removeAfter(List\*& L, int val)

**Mục tiêu:** Xóa node sau node có giá trị val.

**Cách giải:**

Tìm node có key = val.

Xóa node ngay sau nó, nếu có.

#### 10. Hàm addPos(List\*& L, int data, int pos)

**Mục tiêu:** Thêm node tại vị trí pos.

**Cách giải:**

Nếu pos = 0 thì dùng addHead.

Duyệt đến vị trí pos - 1, chèn node mới vào.

#### 11. Hàm removePos(List\*& L, int data, int pos)

**Mục tiêu:** Xóa node tại vị trí pos.

**Cách giải:**

Nếu pos = 0, dùng removeHead.

Duyệt tới pos - 1, bỏ qua node ở pos.

#### 12. Hàm addBefore(List\*& L, int data, int val)

**Mục tiêu:** Thêm node mới trước node có key = val.

**Cách giải:**

Duyệt từng cặp node, nếu next->key == val, chèn vào giữa.

### 13. Hàm addAfter(List\* & L, int data, int val)

**Mục tiêu:** Thêm node sau node có key = val.

**Cách giải:**

Duyệt danh sách, khi gặp node có key = val, chèn node mới vào sau nó.

### 14. Hàm printList(List\* L)

**Mục tiêu:** In toàn bộ danh sách.

**Cách giải:**

Duyệt từng node, in key.

### 15. Hàm countElements(List\* L)

**Mục tiêu:** Đếm số lượng node.

**Cách giải:**

Duyệt từ p\_head, tăng biến đếm.

### 16. Hàm reverseList(List\* L)

**Mục tiêu:** Đảo ngược danh sách.

**Cách giải:**

Duyệt từng node, đảo chiều con trỏ p\_next.

Cập nhật p\_head và p\_tail.

### 17. Hàm removeDuplicate(List\* & L)

**Mục tiêu:** Xóa các node trùng nhau liên tiếp.

**Cách giải:**

Duyệt danh sách, nếu cur->key == cur->p\_next->key, xóa p\_next.

### 18. Hàm removeElement(List\* & L, int key)

**Mục tiêu:** Xóa node có key = key.

**Cách giải:**

Nếu node đầu là key → xóa đầu.

Duyệt danh sách, xóa node có key.

## BÁO CÁO ĐỒ ÁN DOUBLY LINKED

### 1. createNode(int data)

- **Hướng giải:** Tạo node mới, khởi tạo key, pNext, pPrev.

- **Cách giải:** Cấp phát bộ nhớ cho một d\_NODE, gán key = data, con trỏ trước và sau đều nullptr.

## 2. createList(d\_NODE\* p\_node)

- **Hướng giải:** Tạo danh sách mới với node đầu tiên.
- **Cách giải:** Cấp phát một d\_List, gán pHead = pTail = p\_node.

## 3. addHead(d\_List\* & L, int data)

- **Hướng giải:** Thêm node mới vào đầu danh sách.
- **Cách giải:** Nếu danh sách rỗng thì gán pHead = pTail = newNode. Ngược lại, cập nhật liên kết: newNode -> pNext = pHead, pHead -> pPrev = newNode, pHead = newNode.

## 4. addTail(d\_List\* & L, int data)

- **Hướng giải:** Thêm node vào cuối danh sách.
- **Cách giải:** Nếu rỗng thì pHead = pTail = newNode. Ngược lại, gán newNode -> pPrev = pTail, pTail -> pNext = newNode, cập nhật pTail = newNode.

## 5. removeHead(d\_List\* & L)

- **Hướng giải:** Xoá node đầu tiên.
- **Cách giải:** Nếu danh sách có một node thì gán cả pHead và pTail = nullptr. Nếu nhiều hơn thì pHead = pHead->pNext, cập nhật pHead->pPrev = nullptr.

## 6. removeTail(d\_List\* & L)

- **Hướng giải:** Xoá node cuối cùng.
- **Cách giải:** Tương tự như removeHead nhưng thao tác ở pTail.

## 7. removeAll(d\_List\* & L)

- **Hướng giải:** Xoá toàn bộ danh sách.
- **Cách giải:** Dùng vòng lặp gọi removeHead(L) liên tục cho đến khi danh sách rỗng.

## 8. removeBefore(d\_List\* & L, int val)

- **Hướng giải:** Tìm node có giá trị val và xoá node đứng trước nó.
- **Cách giải:** Tìm node current có key == val, nếu current->pPrev != nullptr, thì điều chỉnh con trỏ và xoá toRemove.

## 9. removeAfter(d\_List\* & L, int val)

- **Hướng giải:** Tìm node có giá trị val và xoá node đứng sau.

- **Cách giải:** Tìm current, nếu có current->pNext, thì điều chỉnh con trỏ xoá node kế tiếp.

#### 10. addPos(d\_List\* & L, int data, int pos)

- **Hướng giải:** Thêm node vào vị trí pos cụ thể.
- **Cách giải:** Nếu pos = 0 thì gọi addHead. Ngược lại, duyệt đến vị trí, chèn node vào giữa hoặc cuối danh sách, cập nhật các liên kết.

#### 11. removePos(d\_List\* & L, int data, int pos)

- **Hướng giải:** Xoá node tại vị trí pos nếu key == data.
- **Cách giải:** Duyệt đến vị trí pos, nếu node có key trùng khớp thì xoá bằng cách điều chỉnh liên kết hoặc gọi removeHead, removeTail.

#### 12. addBefore(d\_List\* & L, int data, int val)

- **Hướng giải:** Thêm node trước node có giá trị val.
- **Cách giải:** Nếu pHead->key == val thì gọi addHead. Ngược lại, tìm node val, chèn node mới phía trước.

#### 13. addAfter(d\_List\* & L, int data, int val)

- **Hướng giải:** Thêm node mới sau node có key == val.
- **Cách giải:** Tìm node current, sau đó cập nhật liên kết newNode vào sau current.

#### 14. printList(d\_List\* L)

- **Hướng giải:** In ra toàn bộ danh sách từ đầu đến cuối.
- **Cách giải:** Duyệt từ pHead đến nullptr, in ra key của từng node.

#### 15. countElements(d\_List\* L)

- **Hướng giải:** Đếm số lượng node trong danh sách.
- **Cách giải:** Duyệt danh sách, tăng biến count cho mỗi node.

#### 16. reverseList(d\_List\* L)

- **Hướng giải:** Tạo danh sách mới đảo ngược.
- **Cách giải:** Duyệt từ pHead, thêm từng node mới vào đầu danh sách mới bằng cách thao tác pNext và pPrev ngược.

#### 17. removeDuplicate(d\_List\* & L)

- **Hướng giải:** Xoá các node có giá trị trùng lặp.
- **Cách giải:** Dùng hai vòng lặp (current, runner) để so sánh từng cặp node. Nếu trùng key, xoá runner.

#### 18. removeElement(d\_List\* & L, int key)

- **Hướng giải:** Xoá tất cả các node có giá trị bằng key.
- **Cách giải:** Duyệt toàn danh sách, nếu key khớp, điều chỉnh liên kết và xoá node tương ứng. Trả về true nếu có ít nhất một node bị xoá.

Main: bổ sung so với main linklist

```
d_NODE* current = reversedList16->pHead;
while (current->pNext != nullptr) {
    assert(current->pNext->pPrev == current);
    current = current->pNext;
} // Kiểm tra tính đúng đắn của con trỏ pPrev trong danh sách liên kết đôi sau
    khi bị đảo ngược (reverseList).

assert(list11->pHead->pNext->pPrev == list11->pHead);
// Kiểm tra tính đúng đắn của con trỏ pPrev trong danh sách liên kết đôi.
```


The screenshot shows the GitHub interface for the repository 'lehuynhanhhthu / DSAofathw'. The repository is public and has 0 stars, 0 forks, 1 watching, 1 branch, and 0 tags. The main branch is selected. The file list shows four folders: 'Week1', 'Week3/24120226\_24120247', 'week2', and 'week4'. The 'week4' folder is the most recent, updated 1 minute ago.

File/Folder	Commit Hash	Time Ago
Week1	Week3	2 days ago
Week3/24120226_24120247	Week3	2 days ago
week2	week2	2 weeks ago
week4	week4	1 minute ago

lehuynhanhhthu / DSAofathw

🔍

📁



<> Code

🔗 Issues

🔗 Pull requests

🔗 Actions

🔗 Projects

🔗 Wiki

⋮

← Files


🔗 main

▼

DSAofathw / week4 /

📄

⋮

 lehuynhanhhthu week4

21ce676 · 14 minutes ago

🔄

Name	Last commit message	Last commit date
📁 ..		
📄 DoublyLinkedList.cpp	week4	14 minutes ago
📄 LinkedList.cpp	week4	14 minutes ago