

Singly Linked List

1. Initialize a NODE from a given integer:

• **NODE* createNode(int data)**

- Tạo node có giá trị là data và con trỏ trỏ đến null.

2. Initialize a List from a given NODE:

• **List* createList(NODE* p_node)**

- Tạo list rỗng với structment có sẵn. Gán phần tử p_node đầu vào là head và tail của list.

3. Insert an integer to the head of a given List:

• **bool addHead(List* &L, int data)**

- Tạo Node có giá trị là data bằng hàm createNode.
- Kiểm tra xem có phải List rỗng không, nếu rỗng gán head và tail cho Node.
- Gán con trỏ của Node cho head, gán head = Node.
- Trả về True nếu insert được và false cho trường hợp ngược lại.

4. Insert an integer to the tail of a given List:

• **bool addTail(List* &L, int data)**

- Tạo Node có giá trị là data bằng hàm createNode.
- Kiểm tra xem có phải List rỗng không, nếu rỗng gán head và tail cho Node.
- Gán con trỏ của tail cho Node, gán tail = Node.
- Trả về True nếu insert được và false cho trường hợp ngược lại.

5. Remove the first NODE of a given List:

• **bool removeHead(List* &L)**

- Nếu danh sách rỗng thì trả về false, còn lại trả ra true.
- Gán biết Node temp cho head, gán head = head -> next sau đó giải phóng biến temp.
- Kiểm tra nếu hàm không rỗng thì gán con trỏ tail bằng null.

6. Remove the last NODE of a given List:

• **void removeTail(List* &L)**

- Kiểm tra con trỏ null hoặc là rỗng.
- Nếu chỉ có 1 Node trong List thì xóa node đó, gán con trỏ bằng nullptr rồi trả hàm.
- Dùng biến con trỏ cur để chạy vòng lặp đến kế tiếp node tail.
- Xóa node tail rồi gán node tail mới lại bằng cur, sau đó gán con trỏ về nullptr.

7. Remove all NODE from a given List:

• **void removeAll(List* &L)**

- Chạy vòng lặp với điều kiện L->head != nullptr, rồi dùng hàm removeHead.

8. Remove a Node Before with a given value List:

• **void removeBefore(List* &L, int val)**

- Kiểm tra hàm rỗng hay null ptr, nếu có thì thoát hàm
- Nếu phần tử nằm ở đầu thì gọi lại hàm removeHead(L).

Báo Cáo Thực Hành DSA – Tuần 4

- Gọi Node cur = head, cho chạy vòng lặp với điều kiện “ cur -> next && cur -> next->next đều khác nullptr). Nếu gặp giá trị val thì gọi node temp để delete tương tự như hàm RemoveHead. Trong vòng lặp thì dùng cur = cur -> next để di chuyển con trỏ chạy.

9. Remove an integer after a value of a given List:

• void removeAfter(List* &L, int val)

- Nếu hàm rỗng hay nullptr thì thoát hàm.
- Cho vòng lặp với cur chạy cho tới khi kế tiếp giá trị val thì dừng lại. (cur = cur -> next)
- Nếu giá trị đó không phải giá trị cuối (khác nullptr) thì gọi biến Node temp để lưu tạm thời cur->next , tương tự hàm ở trên, rồi gán biến mới và xóa node temp.

10. Insert an integer at a position of a given List:

• bool addPos(List* &L, int data, int pos)

- Chia trường hợp:
 - o Nếu pos < 0 thì trả ra false vì vị trí trong mảng.
 - o Nếu bằng 0 thì gọi hàm addHead(L,data) vì vị trí đầu.
 - o Nếu >0 thì chạy vòng lặp với biến Node cur = cur -> next cho tới vị trí kế tiếp pos – 1 (chạy với for vì biến đếm pos là giá trị nguyên).
 - Nếu cur lúc này null rồi thì trả ra false (pos > n).
 - Nếu cur lúc này == tail thì gán tail = newNode
 - Còn lại gán như bình thường với trỏ lại con trỏ newNode -> next = cur -> next và giá trị cur -> next = newNode. Con trỏ newNode lúc này tạo ra bởi hàm createNode(data) và trùng với vị trí pos ban đầu.

11. Remove an integer at a position of a given List:

• void RemovePos(List* &L, int data, int pos)

- Chia trường hợp:
 - o Nếu pos < 0 thì trả ra false vì vị trí trong mảng.
 - o Nếu bằng 0 thì gọi hàm RemoveHead(L,data).
 - o Nếu >0 thì chạy vòng lặp với biến Node cur = cur -> next cho tới vị trí kế tiếp pos – 1 (chạy với for vì biến đếm pos là giá trị nguyên).
 - Nếu cur lúc này null rồi thì trả ra false (pos > n).
 - Nếu cur lúc này == tail thì gán tail = cur.
 - Còn lại xóa như bình thường với NODE temp = cur ->next, tương tự và delete temp.

12. Insert an integer before a value of a given List:

• bool addBefore(List* &L, int data, int val)

- Nếu danh sách rỗng thì trả về false.
- Nếu giá trị val là giá trị của head thì gọi hàm addHead(L,data).
- Cho vòng for chạy với cur->next != nullptr là điều kiện, biến ban đầu gán cur = head, và cur = cur ->next. Nếu giá trị biến cur->next->key == val thì thêm giá trị vào tương tự như trên với newNode = createNode(data), trả về true nếu thực hiện được.

13. Insert an integer after a value of a given List:

Báo Cáo Thực Hành DSA – Tuần 4

• **bool addAfter(List* &L, int data, int val)**

- Cho Node cur = head, vòng lặp chạy với điều kiện cur != nullptr và cur = cur -> next. Nếu cur -> key == data thì gán ngay sau nó với biến cur->next. Trường hợp cur == L->tail thì gán L -> tail = newNode. (newNode = createNode(data).)
- Trả ra false nếu không add được.

14. Print all elements of a given List:

• **void printList(List* L)**

- Cho Node cur = head, vòng lặp chạy với điều kiện cur != nullptr và cur = cur -> next. In ra cur -> key và “ “.

15. Count the number of elements List:

• **int countElements(List* L)**

- Cho Node cur = head, vòng lặp chạy với điều kiện cur != nullptr và cur = cur -> next. In ra cur -> key và “ “.
- Tạo biến đếm count++ và trả về để thoát hàm.

16. Create a new List by reverse a given List:

• **List* reverseList(List* L)**

- Chia trường hợp:
 - o Nếu danh sách rỗng thì trả luôn reversedList = createList(nullptr)
 - o Nếu danh sách chỉ có 1 phần tử thì addHead(reversedList, head->key) rồi thoát ra trả list lại.
 - o Tạo biến n = countElement(L) tạo mảng mới với giá trị được lưu của list của for(int i = 0, i < n; i++) rồi gán. Sau đó tạo Node head = createNode(a[0]) rồi addHead(reversedList, a[i]) với i = 1 -> n trong mảng a. Sau đó trả bộ nhớ động mảng a.

17. Remove all duplicates from a given List:

• **void removeDuplicate(List* &L)**

- Nếu danh sách rỗng hoặc chỉ có 1 giá trị thì trả hàm về ban đầu.
- Chạy 2 vòng for, vòng ngoài là cur chạy cho tới nullptr. Chạy vòng for thứ 2 bên trong với biến runner bắt đầu từ cur, điều kiện là runner->next != nullptr. Xét điều kiện if(runner->next -> key == cur->key) thì xóa giá trị (kiểm tra xem có bằng null hay không trước khi xóa).

18. Remove all key value from a given List:

• **bool removeElement(List* &L, int key)**

- Nếu mảng rỗng trả về false
- Nếu key nằm ở đầu thì gọi removeHead(L)
- Cho while(cur->next) với cur từ head nếu next->key == key thì ta gọi lệnh lại remove After(L, cur->key) để xóa hết đồng thời trả ra true.
- Các trường hợp còn lại để trả ra false.

Doubly Linked List

1. Initialize a NODE from a given integer:

• **NODE* createNode(int data)**

- Tạo node có giá trị là data và để 2 con trỏ prev và next đến null.

2. Initialize a List from a given NODE:

• **List* createList(NODE* p_node)**

- Tạo list rỗng với structment có sẵn.
- Gán phần tử p_node đầu vào là head và tail của list, với mỗi node xét next và prev.

3. Insert an integer to the head of a given List:

• **bool addHead(List* &L, int data)**

- Tạo Node có giá trị là data bằng hàm createNode.
- Kiểm tra xem có tạo ra node được không, nếu fail trả ra false
- Kiểm tra xem có phải List rỗng không, nếu rỗng gán head và tail cho Node.
- Gán con trỏ của Node cho head, update head = Node với prev và next.
- Trả về True nếu insert được.

4. Insert an integer to the tail of a given List:

• **bool addTail(List* &L, int data)**

- Tạo Node có giá trị là data bằng hàm createNode.
- Kiểm tra xem có tạo ra node được không, nếu fail trả ra false
- Kiểm tra xem có phải List rỗng không, nếu rỗng gán head và tail cho Node.
- Gán con trỏ của Node cho tail, update head = Node với prev và next.
- Trả về True nếu insert được.

5. Remove the first NODE of a given List:

• **bool removeHead(List* &L)**

- Nếu danh sách rỗng thì trả về false, còn lại trả ra true.
- Gán biết Node temp cho head, gán head = head -> next sau đó giải phóng biến temp.
- Kiểm tra nếu hàm không rỗng thì gán con trỏ tail bằng null.

6. Remove the last NODE of a given List:

• **void removeTail(List* &L)**

- Nếu danh sách rỗng thì trả hàm.
- Gán biết Node temp cho head, gán head = head -> next sau đó giải phóng biến temp.
- Kiểm tra nếu hàm không rỗng thì gán con trỏ tail bằng null.
- Gán tail về prev với tail = tail->prev.

7. Remove all NODE from a given List:

• **void removeAll(List* &L)**

- Chạy từ head rồi xóa hết từ từ,
- Gáp lại head = tail = nullptr

8. Remove a Node Before with a given value List:

• **void removeBefore(List* &L, int val)**

Báo Cáo Thực Hành DSA – Tuần 4

- Kiểm tra danh sách rỗng thì trả ra.
- Chạy Node từ head và kiểm tra $cur, cur \rightarrow next \neq \text{nullptr}$, Nếu gặp $cur \rightarrow next \rightarrow key == val$ thì :
 - o Nếu node là head gán $head = cur \rightarrow next$ và $prev = \text{null}$
 - o Cập nhật vị trí pointer đến vị trí mới (cả $prev$ và $next$), dùng biến trung gian để cập nhật (phải xóa để giải phóng bộ nhớ).

9. Remove an integer after a value of a given List:

• **void removeAfter(List* &L, int val)**

- Kiểm tra danh sách rỗng thì trả ra.
- Chạy Node từ head và kiểm tra $cur, cur \rightarrow next \neq \text{nullptr}$, Nếu gặp $cur \rightarrow next \rightarrow key == val$ thì :
 - o Cập nhật vị trí pointer đến vị trí mới (cả $prev$ và $next$), dùng biến trung gian để cập nhật (phải xóa để giải phóng bộ nhớ).
 - o Nếu node là tail thì cập nhật tail về lại current.

10. Insert an integer at a position of a given List:

• **bool addPos(List* &L, int data, int pos)**

- Chia trường hợp :
 - o Nếu $pos < 0$, trả ra false.
 - o Nếu $pos = 0$ thì $addHead(L, data)$ vì ở đầu. Trả về true.
 - o Nếu $pos > 0$: Tạo Node mới, chạy vòng for để di chuyển vị trí con trỏ đến vị trí của pos . Thêm lệnh kiểm tra pos có hợp lệ hay không. Tạo 1 node mới và cập nhật biến cho $prev$ và $next$. Trả về true

11. Remove an integer at a position of a given List:

• **void RemovePos(List* &L, int data, int pos)**

- Nếu danh sách rỗng hoặc $pos < 0$ thì trả hàm.
- Nếu $pos = 0$ thì gọi $removeHead(L)$.
- Chạy for để di chuyển con trỏ đến vị trí pos . Update con trỏ để xóa biến tại pos (gọi biến cur để lưu giá trị sau đó delete)

12. Insert an integer before a value of a given List:

• **bool addBefore(List* &L, int data, int val)**

- Nếu danh sách rỗng thì trả về false.
- Nếu giá trị val ở head thì gọi hàm $addHead(L, data)$ /
- Cho con trỏ chạy và tìm vị trí target value. Sau đó liên kết node mới (cả $prev$ và $next$) vào danh sách. Thay đổi các biến trỏ đến vị trí đó (Xóa và thêm các con trỏ). Trả về true

13. Insert an integer after a value of a given List:

• **bool addAfter(List* &L, int data, int val)**

- Nếu danh sách rỗng thì trả về false.
- Nếu giá trị val ở head thì gọi hàm $addHead(L, data)$ /

Báo Cáo Thực Hành DSA – Tuần 4

- Cập nhật các giá trị xung quanh node đã tìm thấy để thêm vào danh sách. Kiểm tra xem `cur->next != null` thì gán tail bằng node mới còn không thì cập nhật bình thường. Trả về `true`.

14. Print all elements of a given List:

- **void printList(List* L)**

- Danh sách rỗng thì trả về
- Chạy vòng while và in ra giá trị `cur->key`.

15. Count the number of elements List:

- **int countElements(List* L)**

- Danh sách rỗng thì trả về
- Chạy vòng while và biến đếm ++ cho mỗi là `cur -> next`.

16. Create a new List by reverse a given List:

- **List* reverseList(List* L)**

- Danh sách rỗng thì trả về danh sách mới
- Nếu danh sách có 1 phần tử thì trả về danh sách mới
- Gọi current với thêm node này vào đầu của danh sách mới -> tự đảo ngược. rồi trả về `newList`.

17. Remove all duplicates from a given List:

- **void removeDuplicate(List* &L)**

- Chia trường hợp:
 - o Nếu chỉ có 1 node hoặc rỗng thì trả lại danh sách
 - o Xét từng node trong danh sách, kiểm tra các node trùng với biến runner, chạy tương tự để xóa với lệnh `if` nếu tìm thấy biến cần xóa. Gọi lại hàm `removeAfter` để xóa.



18. Remove all key value from a given List:










- **bool removeElement(List* &L, int key)**








- Nếu danh sách rỗng trả ra `false`
- Chạy vòng lặp từ head của list tới khi gặp giá trị key xét các trường hợp :
 - o Nếu ở đầu thì gọi `removeHead(L)`; trả ra `true`
 - o Nếu ở đuôi thì gọi `removeTail(L)`; trả ra `true`
 - o Nếu ở giữa thì gọi và delete current để giải phóng bộ nhớ; trả ra `true`
- Trả ra `false` nếu không có giá trị nào





Báo Cáo Thực Hành DSA – Tuần 4





https://github.com/BaoNTfit/TH_DSA/tree/main/24120023



 BaoNTfit / TH_DSA











<> **Code**  Issues  Pull requests  Actions  Projects  Wiki  Security 

  main  

 Go to file   Add file 

 **BaoNTfit** Linked-List d961c9d · 1 minute ago 

| Name | Last commit message | Last commit date |
|--|---------------------|------------------|
|  .. | | |
|  .vscode | Linked-List | 1 minute ago |
|  .DS_Store | Linked-List | 1 minute ago |
|  DoublyLinkedList | Linked-List | 1 minute ago |
|  DoublyLinkedList.cpp | Linked-List | 1 minute ago |
|  LinkedList.cpp | Linked-List | 1 minute ago |
|  Singly Linked List.docx | Linked-List | 1 minute ago |
|  ~\$ngly Linked List.docx | Linked-List | 1 minute ago |