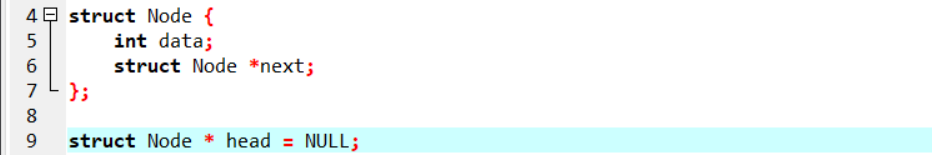
**BÁO CÁO TỔNG HỢP CTDL**

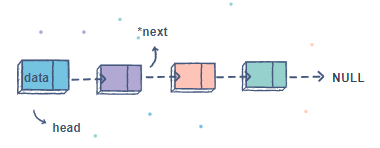
1. **BÀI TẬP TỔNG HỢP LINKED LIST**

**1. BT tại lớp 1 – Singly linked list**

* **Tạo ds liên kết đơn**

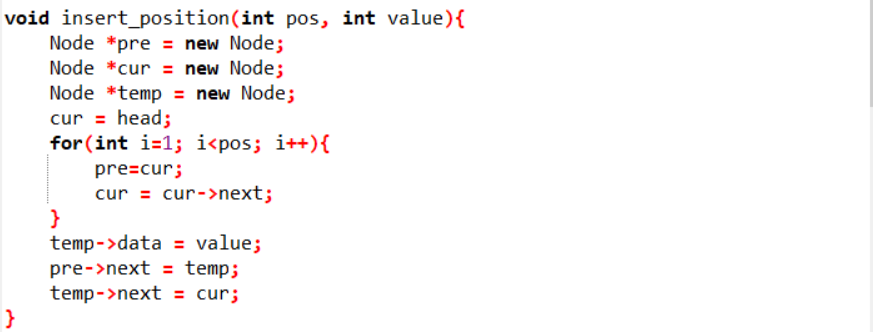
Định nghĩa ra 1 danh sách liên kết đơn bao gồm 2 thành là data chứa giá trị của node hiện tại và con trỏ next để trỏ tới phần tử node tiếp theo.

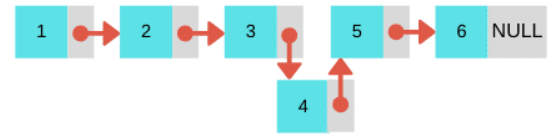
****

****

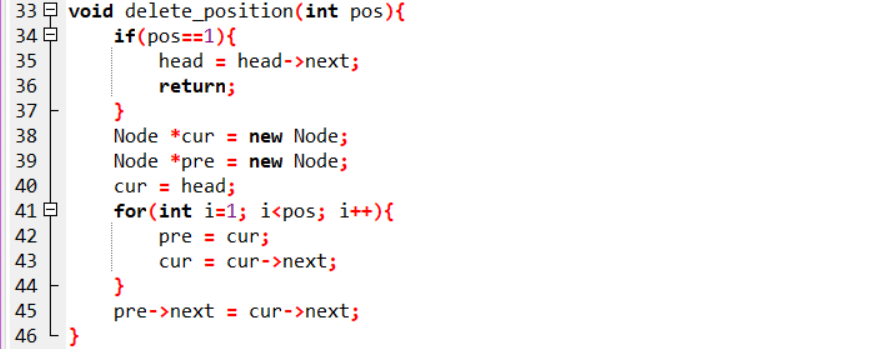
* **Thêm 1 phần tử vào ds lk đơn tại vị trí x**

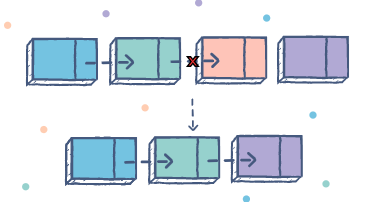
Cách thêm 1 phần tử vào dslk đơn tại vị trí x là dùng 2 con trỏ cur lấy ra phân từ hiện tại và 1 con trỏ pre lấy ra phần từ ngay phía trước. Khi tìm ra được vị trí cần thêm node mới chúng ta sẽ tạo thêm 1 node mới temp. Sau đó cho con trỏ next của pre trỏ đến node mới này và con trỏ next của phần tử mới này trỏ đến node cur.

****

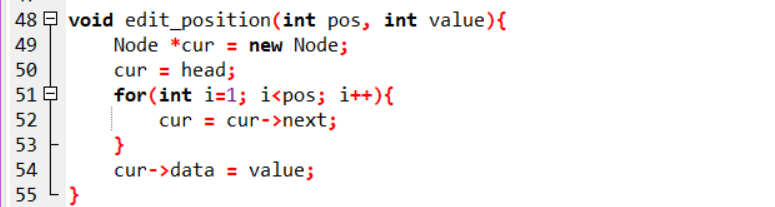
****

* **Xóa 1 phần tử từ ds lk đơn tại vị trí x**

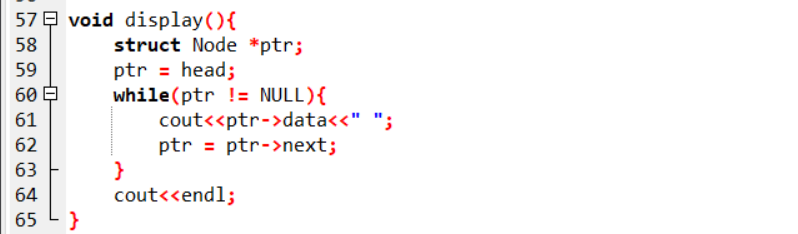
****

****

* **Sửa giá trị cho phần tử tại vị trí x trong dslk đơn**

****

* **Duyệt ds liên kết đơn**

****

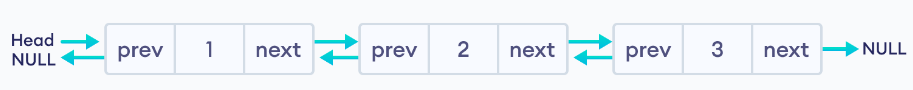
* **Link code full:** [*https://github.com/namlk173/C-/blob/datastructure\_and\_algorithm/LINKED\_LIST.cpp*](https://github.com/namlk173/C-/blob/datastructure_and_algorithm/LINKED_LIST.cpp)

**2. BT tại lớp 2– Doubly linked list**

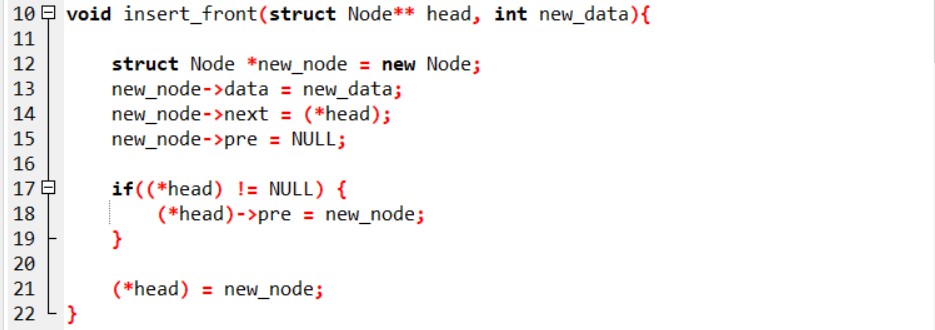
* **Tạo ds liên kết kép**

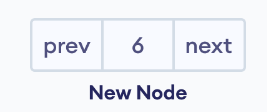
****

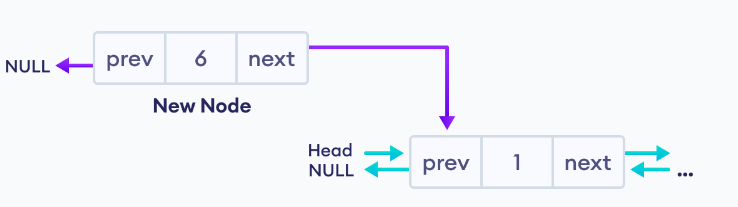
****

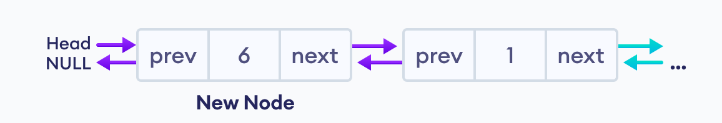
****

* **Thêm 1 phần tử vào đầu ds lk kép**

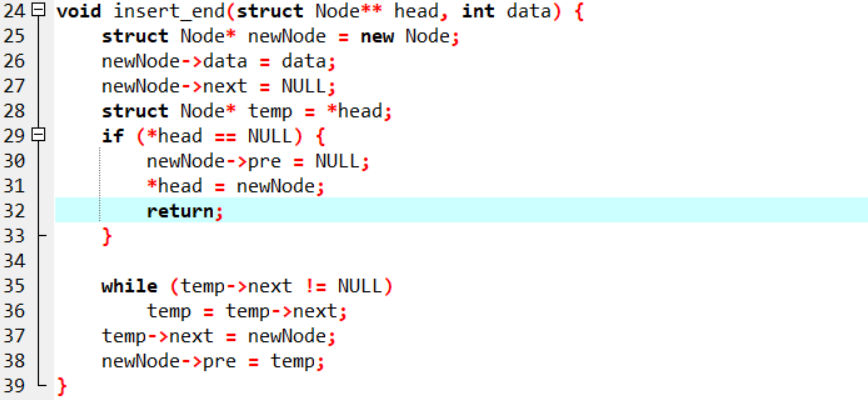
****

****

****

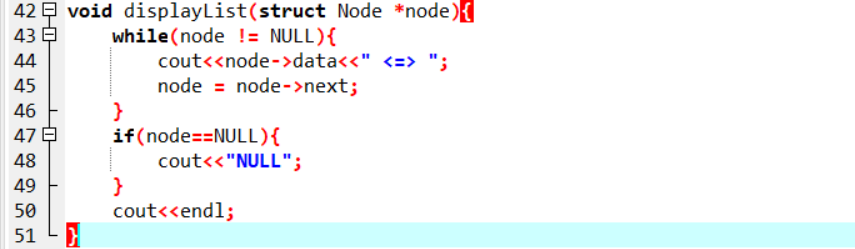
****

* **Thêm 1 phần tử vào cuối ds lk kép**

****

****

* **Duyệt ds liên kết kép**

****

* **Link code full:** [***https://github.com/namlk173/C-/blob/datastructure\_and\_algorithm/Doubly\_List.cpp***](https://github.com/namlk173/C-/blob/datastructure_and_algorithm/Doubly_List.cpp)

**3. BT tại lớp 3: Ứng dụng DSLK**

* **Chọn 1 bài trong code.ptit: Phân tích tính phù hợp và ứng dụng DSLK vào giải quyết bài toán đó.**

1. **BÀI TẬP TỔNG HỢP STACK**

**1. Cài đặt Stack theo 2 cách:**

Thao tác trên Stack:

* Đưa dữ liệu vào Stack
* Lấy dữ liệu ra khỏi Stack

**Dựa vào mảng.**

* **Link code full:**

**Dựa vào danh sách liên kết.**

* **Link code full:**

**2. BT ứng dụng Stack**

**Chọn 1 bài trong code.ptit: Phân tích tính phù hợp và ứng dụng Stack**

**vào giải quyết bài toán đó.**

1. **BÀI TẬP TỔNG HỢP QUEUE**

**1. Cài đặt Queue theo 2 cách:**

**Dựa vào mảng.**

* **Link code full:**

**Dựa vào danh sách liên kết.**

* **Link code full:**

**2. BT ứng dụng Queue: Chọn 1 bài trong code.ptit, phân tích tính phù**

**hợp và ứng dụng Queue vào giải quyết bài toán đó.**

1. **BÀI TẬP TỔNG HỢP BINARY SEARCH TREE**

**1. Cài đặt các thao tác trên cây nhị phân tìm kiếm**

* Tạo node gốc cho cây.
* Thêm vào node vào cây tìm kiếm.
* Loại bỏ node trên cây tìm kiếm.
* Tìm kiếm node trên cây.
* Xoay trái cây tìm kiếm
* Xoay phải cây tìm kiếm
* Duyệt cây theo thứ tự trước.
* Duyệt cây theo thứ tự giữa.
* Duyệt cây theo thứ tự sau.

**2. BT ứng dụng Binary Search Tree: Chọn 1 bài trong code.ptit, phân tích tính**

**phù hợp và ứng dụng Binary Search Tree vào giải quyết bài toán đó.**

1. **BÀI TẬP TỔNG HỢP AVL**

**1. Cài đặt các thao tác trên cây nhị phân tìm kiếm tự cân bằng**

* Xoay trái cây AVL
* Xoay phải cây AVL
* Thêm node vào cây AVL
* Loại bỏ node trên cây AVL

**2. BT ứng dụng AVL: Chọn 1 bài trong code.ptit, phân tích tính phù hợp và**

**ứng dụng AVL vào giải quyết bài toán đó.**