# **Bài 12. Struct và danh sách liên kết**

**Mục tiêu:** *Luyện tập kĩ năng lập trình với struct, con trỏ và bộ nhớ động*

**Yêu cầu nộp bài** *: Toàn bộ code phần A: các câu 1-6 làm chung trong một chương trình (file), các câu còn lại mỗi câu trong một chương trình riêng.*

*Các bài (\*) không bắt buộc, nếu làm bạn sẽ được bù điểm cho bài cũ.*

***Lưu ý: chép bài hoặc cho chép bài sẽ dẫn đến trượt môn học!***

## Thực hành

Danh sách liên kết là cấu trúc dữ liệu quan trọng, là nền tảng của nhiều kiểu dữ liệu trừu tượng khác. Luyện tập cài đặt danh sách liên kết cũng là một cách rất hiệu quả trong việc luyện tập kĩ năng lập trình với con trỏ và bộ nhớ động. Thiết kế và gợi ý thuật toán cũng như một số đoạn code mẫu được cho sẵn trong bài giảng. Nhiệm vụ của bạn là lắp ghép và hoàn thiện hai kiểu dữ liệu danh sách liên kết đơn (singly-linked list), danh sách liên kết đôi (doubly-linked list), kèm theo các hàm xử lý danh sách ở dạng hàm thành viên của các struct.

**Lưu ý:** Để có hiệu quả cao nhất trong việc rèn luyện kỹ năng và tư duy cho bản thân, bạn có thể thảo luận với các sinh viên khác, nhưng không nên tìm kiếm các lời giải có sẵn. Bài này không hề đòi hỏi các thuật toán phức tạp, mà chỉ cần bạn nắm vững các khái niệm và cú pháp đã học. Chính quá trình suy nghĩ tìm cách giải và tìm lỗi sẽ giúp bạn nắm vững các khái niệm con trỏ và bộ nhớ động. Code do người khác viết hoàn toàn không giúp được gì cho bạn, nếu bạn không tự trải qua quá trình đó. Do đó, hãy nỗ lực cá nhân, ít nhất cho đến khi bài mẫu được đăng sau hạn nộp bài.

1. ***addFirst, addLast và print.*** Với định nghĩa SLinkedList với hai con trỏ head và tail, và các đoạn code đã cho trong bài giảng, hãy lắp ghép thành một chương trình với hàm testA1() thử nghiệm việc tạo một danh sách liên kết đơn, dùng addFirst() và addLast() để đẩy dữ liệu dạng xâu kí tự vào trong danh sách, sau mỗi bước lại dùng hàm print() để in kết quả ra màn hình. Gọi hàm testA1() từ hàm main.
2. ***removeFirst.*** Bổ sung hàm removeFirst() vào struct SLinkedList, thử nghiệm hàm đó trong hàm testA2(), chẳng hạn tạo một danh sách, đẩy vào một số xâu kí tự, gọi removeFirst() để xóa dần từ xâu, mỗi lần gọi print() để xem kết quả.   
   Gọi testA2() từ hàm main.
3. ***Destructor.*** Bổ sung hàm destructor cho struct SLinkedList, với nhiệm vụ giải phóng toàn bộ vùng bộ nhớ mà các nút trong danh sách hiện chiếm giữ. Thử nghiệm hàm đó trong hàm testA3(), chẳng hạn cấp phát động một danh sách, đẩy vào một số xâu kí tự, sau đó delete danh sách.   
   Gọi testA3() từ hàm main.
4. ***removeLast.*** Bổ sung hàm removeLast() vào struct SLinkedList với nhiệm vụ xóa nút đứng cuối danh sách, thử nghiệm hàm đó trong hàm testA4(), chẳng hạn tạo một danh sách, đẩy vào một số xâu kí tự, gọi removeLast() để xóa dần từ xâu, mỗi lần gọi print() để xem kết quả.   
   Gọi testA4() từ hàm main.
5. ***insertAfter() cho danh sách liên kết đơn***. Bổ sung hàm insertAfter(Node\* p, const char\* s) vào struct SLinkedList với nhiệm vụ thêm một nút với dữ liệu s vào sau nút p, thử nghiệm hàm đó trong hàm testA5(), chẳng hạn tạo một danh sách, đẩy vào một số xâu kí tự, sau đó gọi insertAfter() để thêm nút mới vào sau phần từ đầu dãy / cuối dãy/ p null..., mỗi lần gọi print() để xem kết quả.   
   Gọi testA5() từ hàm main.
6. ***(\*) remove(Node\*)***. Bổ sung hàm const char\* remove(Node\* p) vào struct SLinkedList với nhiệm vụ xóa nút p khỏi list và trả về dữ liệu lưu trong p, thử nghiệm hàm đó trong hàm testA6().  
   Gọi testA6() từ hàm main.
7. ***Danh sách liên kết đôi***.Tương tự yêu cầu của các bài 1-4, thực hiện cho danh sách liên kết đôi (bổ sung thêm hàm insertAfter()).
8. **(\*) *Kiểm tra tiền điều kiện***. Một số hàm xử lý struct trong hai cấu trúc danh sách trên yêu cầu một số tiền điều kiện, chẳng hạn insertAfter(Node\* p,...) sẽ vô nghĩa nếu p null. Hãy dùng lệnh assert(biểu thức điều kiện) với include <cassert> để các hàm kiểm tra các tiền điều kiện trước khi chạy. Áp dụng cho tất cả các hàm mà bạn nghĩ là cần.