Danh Sách Liên Kết Đơn

Mục tiêu

Mục tiêu của bài thực hành này nhằm giúp các em nắm vững về cấu trúc dữ liệu danh sách liên kết (DSLK) đơn, biết cách vận dụng DSLK đơn vào hiện thực các cấu trúc dữ liệu khác như Stack và Queue, và biết cách áp dụng giải thuật sắp xếp trên cấu trúc dữ liệu DSLK đơn này.

Nội dung

Câu 1 (10đ): Hãy xây dựng một lớp SinglyLinkedList<E> (kiểu Generic) để biểu diễn cấu trúc dữ liệu kiểu danh sách liên kết đơn. Hiện thực các phương thức của lớp:

- int getSize(): cho biết số phần tử đang có trong danh sách
- boolean isEmpty(): cho biết danh sách hiện tại có rỗng hay không ?
- E first(): trả về giá trị của phần tử ở đầu danh sách
- E last(): trả về giá trị của phần tử ở cuối danh sách
- Node<E> search(E e): tìm một Node trong danh sách đang có chứa giá trị e. Kết quả trả về của phương thức này là chính Node đó (nếu tìm được). Nếu Node không tồn tại trong danh sách thì kết quả trả về của phương thức là **NULL**
- void addFirst(E e): thêm một Node mới có giá trị dữ liệu là e vào đầu danh sách
- void addLast(E e): thêm một Node mới có giá trị dữ liệu là e vào cuối danh sách
- void add(E f, E e): thêm một Node mới có giá trị dữ liệu là e vào sau Node có giá trị dữ liệu là f trong danh sách. Nếu Node f không tồn tại thì phần tử mới sẽ được thêm vào cuối danh sách
- void removeFirst(): xóa bỏ Node ở đầu danh sách
- void removeLast(): xóa bỏ Node ở cuối danh sách
- void remove(E e): xóa bỏ Node đầu tiên có giá trị là e trong danh sách
- void removeAll(E e): xóa bỏ tất cả các Node có giá trị là e trong danh sách
- void swapNode(Node<E> nodeA, Node<E> nodeB): hoán vị/hoán đổi vị trí của hai Node A và B trong danh sách
- swapValue(E a, E b): hoán vị/ hoán đổi giá trị của hai Node đang có chứa giá trị lần lượt là a và b trong danh sách. Nếu không đồng thời tìm ra được 2 Node như vậy trong danh sách thì phương thức này không cần thực hiện.

Ghi chú: Sinh viên nộp bài tập này cho Giảng viên theo hướng dẫn

Câu 2: Viết chương trình nhập vào một danh sách liên kết đơn có các phần tử như hình vẽ dưới đây:

a/ Hãy thực hiện giải thuật **tìm kiếm tuyến tính** (Linear Search) trên danh sách liên kết đơn này để xác định vị trí của phần tử có giá trị x = -28 trong danh sách. Xuất ra vị trí của phần tử tìm được hoặc -1 nếu không tìm được.

b/ Hãy thực hiện **giải thuật sắp xếp nổi bọt** (Buble Sort) cho các phần tử trong danh sách trên theo thứ tự tăng dần

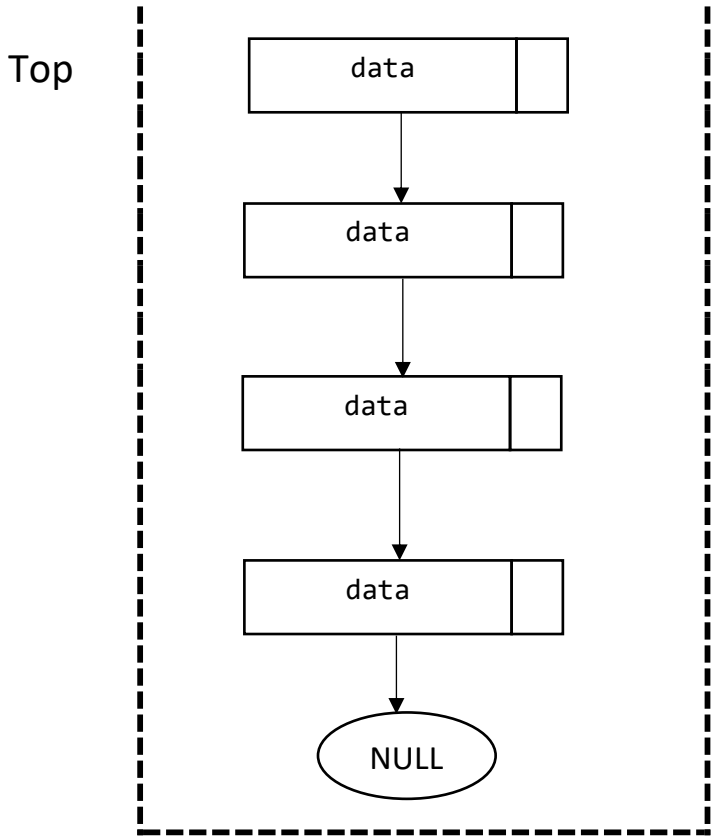
c/ Hãy thực hiện **giải thuật sắp xếp nhanh** (Quick Sort) cho các phần tử trong danh sách trên theo thứ tự giảm dần

d/ Hãy thực hiện **giải thuật tìm kiếm nhị phân** (Binary Search) trên danh sách liên kết đơn này sau khi đã được sắp xếp ở **câu b** hoặc **c** để tìm vị trí của phần tử có giá trị x = 14 trong danh sách. Xuất ra vị trí của phần tử tìm được hoặc -1 nếu không tìm được.

Câu 3: Xây dựng một lớp "Ngăn Xếp Danh Sách Liên Kết Đơn" có tên SinglyLinkedStack<E> dựa trên danh sách liên kết đơn đã xây dựng ở **Câu 1** với các phương thức như sau:

- boolean empty(): kiểm tra xem SinglyLinkedStack <E> có rỗng hay không
- E peek(): trả về giá trị của phần tử ở **đỉnh** SinglyLinkedStack <E> nhưng không lấy phần tử này ra khỏi ngăn xếp, trả về giá trị NULL nếu hàng đợi không có phần tử nào
- E pop(): trả về giá trị của phần tử ở **đỉnh** SinglyLinkedStack <E> và xóa phần tử này khỏi SinglyLinkedStack <E> (lấy phần tử ở đỉnh ra khỏi SinglyLinkedStack <E>) , trả về giá trị NULL nếu hàng đợi không có phần tử nào
- void push(E e): thêm một phần tử có giá trị là e vào đỉnh SinglyLinkedStack <E>

Minh họa SinglyLinkedStack <E> với DSLK đơn:



Câu 4: Xây dựng một lớp "Hàng Đợi Danh Sách Liên Kết Đơn" có tên SinglyLinkedQueue<E> dựa trên danh sách liên kết đơn đã xây dựng ở **Câu 1** với các phương thức như sau:

- boolean empty(): kiểm tra xem SinglyLinkedQueue <E> có rỗng hay không
- E peek(): trả về giá trị của phần tử ở **đầu** SinglyLinkedQueue <E> nhưng không lấy phần tử này ra khỏi hàng đợi, trả về giá trị NULL nếu hàng đợi không có phần tử nào
- E poll(): trả về giá trị của phần tử ở **đầu** SinglyLinkedQueue <E> và xóa phần tử này khỏi SinglyLinkedQueue <E> (lấy phần tử ở đầu ra khỏi SinglyLinkedQueue)
- void add(E e): thêm một phần tử có giá trị là e vào **đuôi** SinglyLinkedQueue <E>

Minh họa SinglyLinkedQueue <E> với DSLK đơn: