***Câu 1: Trình bày từng bước các thao tác trên danh sách liên kết đơn***

- Danh sách liên kết đơn là một cấu trúc dữ liệu được sử dụng rộng rãi trong lập trình để lưu trữ và quản lý các phần tử dưới dạng liên kết các nút. Mỗi nút trong danh sách liên kết đơn chứa một giá trị và một tham chiếu đến nút tiếp theo trong danh sách.

- Các bước để thực hiện các thao tác trên danh sách liên kết đơn:  
 *+ Khởi tạo danh sách liên kết đơn:*  
 • Tạo một nút đầu tiên và đặt giá trị của nó.  
 • Tham chiếu của nút đầu tiên sẽ được đặt là null để chỉ ra rằng danh sách đang trống.

|  |
| --- |
| //Khai báo node  struct Node {  int Data;  Node\* Next;  };  //Khai báo list  struct List {  Node\* pHead;  Node\* pTail;  }; |

*+ Thêm một phần tử vào danh sách:*  
 • Tạo một nút mới chứa giá trị của phần tử muốn thêm.  
 • Tham chiếu của nút mới sẽ được đặt là null.  
 • Tìm nút cuối cùng trong danh sách và đặt tham chiếu của nút cuối cùng đến nút mới.

|  |
| --- |
| //Thêm vào đầu danh sách  void AddpHead(List& l, Node\* p) {  if (l.pHead == NULL) {  l.pHead = p;  l.pTail = l.pHead;  }  else {  p->Next = l.pHead;  l.pHead = p;  }  }  //Thêm vào cuối danh sách  void AddpTail(List& l, Node\* p) {  if (l.pHead == NULL) {  l.pHead = p;  l.pTail = l.pHead;  }  else {  l.pTail->Next = p;  l.pTail = p;  }  }  //Thêm p vào sau node q cho trước  void InsertAfter(List& l, Node\* p, Node\* q) {  if (q != NULL) {  p->Next = q->Next;  q->Next = p;  if (l.pTail == q)  l.pTail = p;  }  else {  AddpHead(l, p); //Thêm vào đầu danh sách  }  } |

*+ Xóa một phần tử khỏi danh sách:* • Tìm nút chứa phần tử muốn xóa.  
 • Đặt tham chiếu của nút trước nó đến nút tiếp theo của nó để bỏ qua nút đó.  
 • Xóa nút chứa phần tử muốn xóa.

*+ Tìm một phần tử trong danh sách:* • Bắt đầu từ nút đầu tiên, duyệt qua từng nút trong danh sách liên kết đơn và so sánh giá trị của nút đó với giá trị muốn tìm.  
 • Nếu giá trị muốn tìm khớp với giá trị của một nút, trả về nút đó.  
 • Nếu không tìm thấy phần tử muốn tìm, trả về null hoặc một giá trị đặc biệt để chỉ ra rằng phần tử đó không có trong danh sách.

|  |
| --- |
| //Tìm một phần tử trong danh sách  Node\* Search(List l, int x) {  Node\* p;  p = l.pHead;  while ((p != NULL) && (p->Data != x))  p = p->Next;  return p;  } |

*+ Sắp xếp danh sách:* • Duyệt từng phần tử trong danh sách và so sánh giá trị của chúng để sắp xếp lại theo thứ tự tăng dần hoặc giảm dần.  
 • Có nhiều cách để sắp xếp danh sách, nhưng thường sử dụng các thuật toán sắp xếp.

|  |
| --- |
| //Sắp xếp trực tiếp  void SelectionSort\_Increase(List& l) {  Node\* p, \* q, \* min;  p = l.pHead;  while (p != l.pTail) {  min = p;  q = p->Next;  while (q != NULL) {  if (q->Data < p->Data)  min = q;  q = q->Next;  }  swap(min->Data, p->Data);  p = p->Next;  }  } |