Trường: **ĐH CNTP TP.HCM** Khoa: **Công nghệ thông tin** Bộ môn: **Công nghệ phần mềm.** MH: **TH Cấu trúc dữ liệu & giải thuật**	**BÀI 1. DANH SÁCH LIÊN KẾT ĐƠN**	

A. MỤC TIÊU:

- Trình bày được hoạt động của danh sách liên kết đơn: thêm, xóa, tìm kiếm
- Xây dựng được danh sách liên kết đơn với các thao tác cơ bản như: khởi tạo, thêm, tìm kiếm, xử lý tính toán, xuất nội dung ra màn hình.
- Hoàn thành được các bài tập tại lớp, tự làm được các bài tập về nhà

B. DỤNG CỤ - THIẾT BỊ THÍ NGHIỆM CHO MỘT SV:

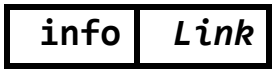
STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

C. NỘI DUNG THỰC HÀNH

I. Tóm tắt lý thuyết

1.1. Khái niệm nút

1.2.1. Cấu tạo nút



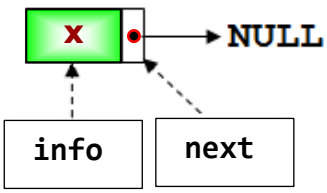
- Thành phần dữ liệu (***info***): Lưu trữ các thông tin về bản thân phần tử.
- Thành phần mối liên kết (***Link***): Lưu trữ địa chỉ của phần tử kế tiếp trong danh sách, hoặc lưu trữ giá trị **NULL** nếu là phần tử cuối danh sách.

1.2.2. Khai báo nút

```
struct Node //Single Node
{
    <Data> info; //Lưu thông tin của nút hiện hành. Data là kiểu
                 //dữ liệu của node.
                 //Giả sử trong bài, node đang chứa số int
    Node* next; //Con trỏ chỉ đến nút kế sau.
};
```

1.2.3. Tạo nút chứa giá trị x

```
Node* createNode(int x)
{
```



```
    Node* p = new Node;
    if(p == NULL)
    {
       printf("Không đủ bộ nhớ để cấp phát!");
       getch();
       return NULL;
    }
    p→info = x;
    p→next = NULL;
    return p;
}
```

1.2.4. Xuất nội dung của nút

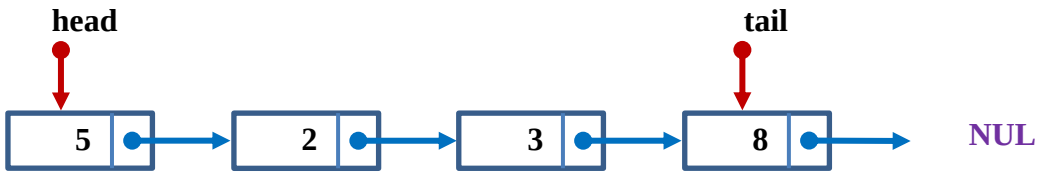
```
void showNode(Node* p)
{
    printf("%d ", p→info);
}
```

1.2. Danh sách liên kết đơn

1.2.1. Khái niệm danh sách liên kết đơn

– Danh sách liên kết đơn gồm 2 con trỏ head và tail. Con trỏ head trỏ đến nút đầu tiên của danh sách, và con trỏ tail trỏ đến nút cuối cùng của danh sách.

– Mỗi phần tử liên kết với phần tử đứng kế sau nó trong danh sách, phần tử cuối trỏ **NULL**.



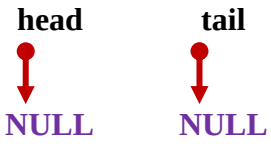
– Khai báo kiểu dữ liệu SList để lưu trữ danh sách liên kết đơn, sử dụng 2 con trỏ head để quản lý nút đầu và tail để quản lý nút cuối:

```
struct SList //dslk đơn – Single List
{
    //Định nghĩa kiểu dữ liệu cho DSLK đơn SList
    Node* head; //Lưu địa chỉ nút đầu tiên trong SList
    Node* tail; //Lưu địa chỉ của nút cuối cùng trong SList
};
```

1.2.2. Một số phương thức trên danh sách liên kết đơn

a. Khởi tạo danh sách

Do danh sách chưa có phần tử nào, nên con trỏ đầu head (*lưu địa chỉ của nút đầu tiên*) và con trỏ cuối tail (*lưu địa chỉ của nút cuối cùng*) đều bằng NULL.



b. Kiểm tra danh sách rỗng

Kiểm tra danh sách liên kết đơn đã có phần tử nào hay chưa? Nghĩa là kiểm tra con trỏ đầu head (*hoặc con trỏ cuối tail*) có bằng NULL hay không? Hàm trả sẽ về 1 nếu danh sách liên kết đơn chưa có phần tử nào (*rỗng*), ngược lại (*không rỗng*) thì hàm sẽ trả về 0.

c. Duyệt danh sách để xuất nội dung ra màn hình

- **Bước 1:** Nếu danh sách rỗng thì: Thông báo danh sách rỗng và không thực hiện.
- **Bước 2:** Gán p = head; *//p lưu địa chỉ của phần tử đầu trong SList*
- **Bước 3:**

  Lặp lại trong khi (con trỏ p còn khác NULL) thì thực hiện:

  + Xuất nội dung của phần tử tại con trỏ p.

  + p = p→next; *//Xét phần tử kế*

d. Thêm nút p có giá trị x vào đầu danh sách.

- **Bước 1:** Nếu phần tử muốn thêm p không tồn tại thì không thực hiện.
- **Bước 2:** Nếu SList rỗng thì

  + head = p;

  + tail = p;
- **Bước 3:** Ngược lại

  + p→next = head;

  + head = p;

e. Thêm nút p có giá trị x vào cuối danh sách.

- **Bước 1:** Nếu phần tử muốn thêm p không tồn tại thì không thực hiện.
- **Bước 2:** Nếu SList rỗng thì:

  + head = p;

  + tail = p;
- **Bước 3:** Ngược lại

  + tail→next = p;

+ tail = p;

f. Thêm nút p có giá trị x vào sau nút q có giá trị y của danh sách.

- **Bước 1:** Nếu phần tử q hoặc phần tử muốn thêm p không tồn tại thì không thực hiện.
- **Bước 2:**
  + p→next = q→next;
  + q→next = p;
  + Nếu q là phần tử cuối thì: tail = p;

g. Tìm kiếm trong danh sách có nút p nào chứa giá trị x.

- **Bước 1:** Nếu danh sách rỗng thì: trả về NULL;
- **Bước 2:** Gán p = head; *//địa chỉ của phần tử đầu tiên trong DSLK đơn*
- **Bước 3:**
  Lặp lại trong khi (p != NULL và p→Info != x) thì thực hiện:
  p = p→next; *//xét phần tử kế sau*
- **Bước 4:** Trả về p; *//nếu (p != NULL) thì p lưu địa chỉ của phần tử có khóa bằng x, hoặc NULL là không có phần tử cần tìm.*

h. Tạo danh sách (nhập giá trị từ bàn phím)

```
void createSList(SList &sl)
{
    int n;
    int x;
    initSList(sl);
    do
    {
       printf("Cho biet so phan tu cua danh sach (n>0): ");
       scanf("%d", &n);
    }while(n <= 0);
    for(int i = 1; i <= n; i++)
    {
       printf("Nhập phần tử thứ %d là: ", i);
       scanf("%d", &x);
       Node* p = createNode(x);
       insertTail(sl, p); //Hoặc chèn vào đầu Inserthead(sl, p);
```

```
        }
    }
```

i. Tạo danh sách tự động

```
void createAutoSList(SList &sl)
{
    int n;
    int x;
    initSList(sl);
    do
    {
        printf("Cho biet so phan tu cua danh sach (n>0): ");
        scanf("%d", &n);
    } while(n <= 0);
    srand(time(NULL)); //Thư viện stdlib.h và time.h
    for(int i = 1; i <= n; i++)
    {
        //Tạo 1 số ngẫu nhiên trong đoạn [-99, 99]
        x = (rand()%199)-99;
        SNode* p = CreateNode(x);
        //Hoặc chèn vào đầu Inserthead(sl, p);
        insertTail(sl, p);
    }
}
```

II. Bài tập hướng dẫn mẫu

Bài 1: Xây dựng một danh sách liên kết đơn chứa các số nguyên.

- Cài đặt các phương thức thêm đầu, thêm cuối và thêm sau một phần tử vào DSLK
- Cài đặt phương thức cho biết phần tử x có tồn tại trong danh sách hay không?
- Cài đặt phương thức xóa phân tử của DSLK: xóa đầu, xóa sau, xóa phần tử sau node q.
- Cài đặt phương thức xuất danh sách ra màn hình
- Xây dựng hàm main (dạng menu) minh họa

Bước 1: Tạo một Project mới.

Bước 2: Khai báo các thư viện cơ bản cho chương trình.

Bước 3: Định nghĩa các kiểu dữ liệu: SNode, SList

Bước 4: Viết các hàm cơ bản xử lý nút:

```
//================================================================
```

- Viết hàm tạo mới (cấp phát) một nút lưu trữ giá trị là x.
- Viết hàm hiển thị thông tin của một nút.
- Viết hàm hiển thị menu của chương trình.
- Viết hàm cho phép chọn bài tương ứng muốn thực thi, chỉ thoát chương trình khi nhập chọn=0.

III. Bài tập ở lớp

Bài 1. Sinh viên thực hiện lại Bài tập mẫu. Sau đó hãy bổ sung thêm các chức năng sau:

a. Thêm phần tử mới vào cuối danh sách ***sl***.
b. Chèn thêm phần tử có giá trị **x** vào trước phần tử có giá trị y.
c. Viết hàm xóa các phần tử lớn hơn x trong dslk
d. Viết hàm xóa các phần tử chẵn trong dslk
e. Sắp xếp dslk tăng dần, giảm dần
f. Cho biết trong dslk có bao nhiêu số nguyên tố
g. Tính tổng các số chính phương trong dslk
h. Tìm phần tử nhỏ nhất, phần tử lớn nhất trong dslk
i. Cho biết trong dskl có bao nhiêu phần tử lớn hơn gấp đôi phần tử phía sau nó.
j. Từ ***sl*** tạo 2 danh sách mới: ***sl1*** chứa các số chẵn, ***sl2*** chứa các số lẻ.

Hướng dẫn:

+ Khởi tạo 2 danh sách sl1 và sl2 rỗng.
+ Lặp lại trong khi sl còn phần tử, kiểm tra:
  - Nếu phần tử có giá trị chẵn thì thêm vào cuối sl1.
  - Nếu phần tử có giá trị lẻ thì thêm vào cuối sl2.

Bài 2. Cho danh sách đơn ***sl*** chứa các phân số, biết mỗi phân số có 2 thành phần là tử số (TuSo) và mẫu số (MauSo). Hãy viết chương trình cho phép:

a. Tạo cấu trúc dữ liệu ***PhanSo***, ***SList***.
b. Nhập/xuất danh sách có **n** phân số.
c. Tối giản các phân số.
d. Tính tổng/tích các phân số.
e. Cho biết phân số lớn nhất, phân số nhỏ nhất.
f. Tăng mỗi phân số của danh sách lên 1 đơn vị.
g. Xuất các phân số lớn hơn 1 trong danh sách liên kết
h. Viết hàm trả về SNode chứa phân số p trong danh sách liên kết. Nếu không có p trong DSLK thì trả về NULL

IV. Bài tập về nhà

Bài 3. Giả sử có một danh sách đơn ***sl*** chứa các số nguyên đã có thứ tự tăng. Cài đặt các hàm sau:

a. Chèn phần tử **x** vào danh sách sao cho vẫn giữ nguyên thứ tự tăng.

b. In danh sách trên theo thứ tự giảm dần.

c. Nối danh sách ***sl2*** vào sau phần tử có giá trị **x** trong danh sách ***sl1*** – nếu không có phần tử **x** thì thông báo "không có phần tử **x**" với sl2 là danh sách chứa các số nguyên

Bài 4. Cho 2 danh sách đơn chứa các số nguyên là ***sl1*** và ***sl2*** có thứ tự tăng. Cài đặt các hàm sau:

a. Trộn 2 danh sách trên thành một danh sách ***sl*** có thứ tự tăng.

Hướng dẫn:

+ Khởi tạo danh sách sl rỗng.
+ Lặp lại trong khi sl1 và sl2 vẫn còn phần tử, kiểm tra:
    - Nếu phần tử của sl1 có giá trị nhỏ hơn hoặc bằng phần tử của sl2 thì thêm phần tử này vào cuối sl.
    - Ngược lại: Nếu phần tử của sl2 có giá trị nhỏ hơn phần tử của sl1 thì thêm phần tử này vào cuối sl.
+ Nếu danh sách nào còn phần tử thì thêm tất cả các phần tử đó vào cuối sl.

b. Trộn 2 danh sách trên thành một danh sách sl có thứ tự giảm.

c. Trộn 2 danh sách trên thành một danh sách sl sao cho các phần tử có giá trị chẵn tăng dần, các phần tử có giá trị lẻ giảm dần.

d. Trộn 2 danh sách trên thành một danh sách ***sl*** sao cho các phần tử tại những vị trí chẵn tăng dần, các phần tử tại những vị trí lẻ giảm dần.

--- HẾT ---