

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ SÀI GÒN  
KHOA CÔNG NGHỆ THÔNG TIN

Giáo trình thực hành  
**NHẬP MÔN**  
**CẤU TRÚC DỮ LIỆU**

Họ và tên ..... Ngày sinh .....

Mã số sinh viên ..... Lớp .....

Bậc ..... Khoa .....

Điện thoại: ..... Email .....

LUU HÀNH NỘI BỘ  
Năm 2020

## **MỤC LỤC**

Bài 1: Làm quen với môi trường Visual C++ 2013    1

Bài 2: Các cấu trúc điều khiển

Bài 3: Các cấu trúc lặp

Bài 4: Cấu trúc lặp do...while

Bài 5: Hàm – cơ chế truyền tham trị

Bài 6: Hàm – cơ chế truyền tham biến

Bài 7: Mảng một chiều

Bài 8: Mảng hai chiều

Bài 9: Xâu kí tự

Bài 10: Ôn tập

## BÀI 1: ÔN TẬP MẢNG, CẤU TRÚC - GIẢI THUẬT TÌM KIẾM


### I. MỤC TIÊU:

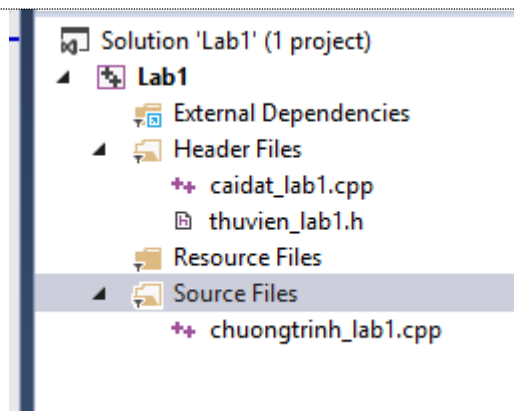
Trong bài thực hành này, sinh viên sẽ:

- Ôn tập lại cách khai báo và truy cập mảng một chiều.
- Áp dụng cấu trúc mảng để cài đặt các giải thuật tìm kiếm.

### II. TÓM TẮT LÝ THUYẾT:

Nội dung	Ví dụ
<b>1. Ôn tập phần mảng</b>	
<b>Khai báo:</b> <b>&lt;kiểu dữ liệu&gt; &lt;tên mảng&gt;[&lt;số phần tử mảng&gt;]</b> <i>Lưu ý:</i> Các phần tử của mảng sẽ được tính từ 0. Vì vậy, phần tử đầu tiên trong mảng có chỉ số là 0, phần tử cuối cùng có chỉ số là <số phần tử mảng -1>.	<pre>int test[5];</pre> <i>// khai báo mảng test có tối đa 5 phần tử, mỗi phần tử có kiểu int.</i> <i>// phần tử đầu tiên có chỉ số là 0, phần tử cuối cùng có chỉ số là 4.</i>
<b>Truy xuất:</b> các phần tử mảng theo cú pháp: <b>&lt;tên mảng&gt;[&lt;chỉ số phần tử&gt;]</b>	Trong khai báo mảng test ở ví dụ trên: <pre>test[0] // truy xuất giá trị phần tử đầu tiên</pre> <pre>test[1] // truy xuất giá trị phần tử thứ hai.</pre> <pre>test[4] // truy xuất giá trị phần tử cuối cùng</pre>
<b>Nhập giá trị:</b> cho từng phần tử mảng	<pre>cin&gt;&gt;test[0]; //nhập giá trị cho phần tử đầu.</pre> <pre>cin&gt;&gt;test[4]; //nhập giá trị cho phần tử cuối cùng trong mảng test</pre> Ngoài ra ta có thể thực hiện gán cho từng giá trị như sau: <pre>test [0] = 5 ;</pre> <pre>test [1] = -10 ;</pre>
<b>2. Tìm kiếm:</b> Tùy theo dữ liệu đã được sắp xếp hay chưa mà ta có 2 phương pháp tìm kiếm: tuyến tính (tuần tự) và nhị phân. Dưới đây là đoạn mã cài đặt giải thuật tìm kiếm tuần tự và tìm kiếm nhị phân.	
<b>a. Tìm kiếm tuần tự</b>	<pre>int timKiem_TuanTu(int a[], int n, int x){</pre> <pre>    int i = 0;</pre>

	<pre> while(i&lt;n &amp;&amp; a[i]!=x)     i++; if (i == n)     return -1; return i; } </pre>
<b>b. Tìm kiếm nhị phân:</b>	<pre> int    <b>timKiem_NhiPhan</b> (int a[], int n, int x){     int bottom=0, top= n -1; int mid;     do{         mid = (bottom + top)/2;         if ( a[mid]==x)             return mid;         else             if(x&lt;a[mid])                 top=mid-1;             else                 bottom=mid+1;     } while (bottom &lt;= top);     return -1; } </pre>
<p><b>GHI CHÚ:</b></p> <p>Trong môn học này, chúng ta sẽ thống nhất trong một project sẽ tiến hành cài đặt theo 3 file: gồm có: <code>thuvien_lab_n</code>, <code>caidat_lab_n</code>, <code>chuongtrinh_lab_n</code>.</p> <p><b>Lưu ý:</b> n ở đây là thứ tự các bài lab</p> <ul style="list-style-type: none"> <li>- <b><code>thuvien_lab_n</code></b>: Chứa các thư viện cần khai báo của chương trình</li> <li>- <b><code>caidat_lab_n</code></b>: Chứa các hàm cài đặt của chương trình</li> <li>- <b><code>chuongtrinh_lab_n</code></b>: Chứa void main() để chạy chương trình</li> </ul> <p> Ví dụ về cách tạo 3 file trong một chương trình</p> <ul style="list-style-type: none"> <li>- <b>Tạo một project ban đầu:</b></li> </ul>	



### thuvien\_lab1.h

```
thuvien_lab1.h*  caidat_lab1.cpp*  chuongtrinh_lab1.cpp
Lab1 (Global Scope)
#include <iostream>
#include <iomanip>
#define KTM 10
using namespace std;
//Khai báo các hàm sẽ sử dụng trong chương trình
void nhapMang(int b[], int &);
void xuatMang(int b[], int);
```

### caidat\_lab1.cpp

```
thuvien_lab1.h*  caidat_lab1.cpp*  chuongtrinh_lab1.cpp
Lab1 (Global Scope)
#include "thuvien_lab1.h"
//Định nghĩa các hàm sẽ xây dựng
void nhapMang(int b[], int &m){
}
void xuatMang(int b[], int){
}
```

### chuongtrinh\_lab1.cpp

```
thuvien_lab1.h*  caidat_lab1.cpp*  chuongtrinh_lab1.cpp*
Lab1 (Global Scope)
#include "thuvien_lab1.h"
void main(){
//Khai báo biến và thực hiện gọi các hàm sẽ thực thi
int a[KTM], n, x;
nhapMang(a, n);
xuatMang(a, n);
}
```

### III. NỘI DUNG THỰC HÀNH:

1. Sinh viên nhập đoạn mã minh họa của lab1\_a:

**Câu 1:** Đoạn mã dưới đây dùng để nhập vào một mảng, xuất ra mảng vừa nhập. Yêu cầu sinh viên nhập đoạn mã sau theo cơ chế 3 file:

- File **thuvien\_lab1.h** như sau:

Mã lệnh
<pre>#include &lt;iostream&gt; #include &lt;iomanip&gt; #define KTM 10 using namespace std; //Khai báo các hàm sẽ sử dụng trong chương trình void nhapMang(int b[], int &amp;); void xuatMang(int b[], int);</pre>

- File **caidat\_lab1.cpp** như sau:

Mã lệnh
<pre>#include "thuvien_lab1.h" //Định nghĩa các hàm sẽ xây dựng void nhapMang(int b[], int &amp;m){     do{         cout &lt;&lt; "\nNhap so phan tu mang:";         cin &gt;&gt; m;     } while (m &lt;= 0    m&gt;KTM);     cout &lt;&lt; "\nNhap gia tri cho mang:";     for (int i = 0; i&lt;m; i++){         cout &lt;&lt; "\nNhap phan tu thu :" &lt;&lt; i &lt;&lt; ": ";         cin &gt;&gt; b[i];     } }  void xuatMang(int b[], int n){     cout &lt;&lt; "\nMang la:";     for (int i = 0; i&lt;n; i++)         cout &lt;&lt; b[i] &lt;&lt; setw(3);</pre>

```
}
|
```

- File `chuongtrinh_lab1.cpp` như sau:

Mã lệnh
<pre>#include"thuvien_lab1.h" void main(){ //Khai bao bien va thuc hien goi cac ham se thuc thi     int a[KTM], n;     nhapMang(a, n);     xuatMang(a, n); }</pre>

**Yêu cầu:**

- Viết đoạn mã trên và biên dịch.
- Nếu khi chạy chương trình ta nhập số phần tử là **12** thì chương trình có chạy không? Tại sao?

**Câu 2:** Sinh viên được cung cấp đoạn mã của hàm nhập mảng, xuất mảng theo một cách khác. Dưới đây là đoạn mã minh họa:

Mã lệnh
<pre>void nhapMang(int b[], int &amp;m){     cout &lt;&lt; "\nNhap gia tri cho mang:";     for (int i = 0; i&lt;m; i++){         cout &lt;&lt; "\nNhap phan tu thu :" &lt;&lt; i &lt;&lt; ": ";         cin &gt;&gt; b[i];     } } void xuatMang(int b[], int n){     cout &lt;&lt; "\nMang la:";     for (int i = 0; i&lt;n; i++)         cout &lt;&lt; b[i] &lt;&lt; setw(3); }</pre>

**Yêu cầu:**

- So sánh sự khác nhau giữa 2 hàm nhập mảng ở **câu 1** và **câu 2**.

b/. Mở file `chuongtrinh_lab1.cpp` để gọi lại hàm nhập mảng vừa viết. (Viết thêm nhập số phần tử cho mảng)

**Câu 3:** Sinh viên bổ sung vào chương trình đã viết thuật toán tìm kiếm tuần tự và gọi hàm đã cho dưới đây:

- File **`caidat_lab1.cpp`** nội dung hàm tìm kiếm tuần tự:

**Mã lệnh**

```
int timKiem_TuanTu(int a[], int n, int x){
    int i = 0;
    while (i<n && a[i] != x)
        i++;
    if (i == n)
        return -1;
    return i;
}
```

- File **`chuongtrinh_lab1.cpp`** như sau:

**Mã lệnh**

```
#include"thuvien_lab1.h"
void main(){
    //Khai bao bien va thuc hien goi cac ham se thuc thi
    int a[KTM], n,x;
    nhapMang(a, n);
    xuatMang(a, n);
    cout << "\n Nhập phần tử cần tìm:";
    cin >> x;
    if (timKiem_TuanTu(a, n, x) >= 0)
        cout << "\n tìm thấy " << x;
    else
        cout << "\n Không tìm thấy " << x;
}
```

**Yêu cầu:**

- a/. Ở trong hàm TKTT ở trên, giải thích giá trị trả về -1.
- b/. Viết thêm vào trong hàm main để xuất ra vị trí tìm thấy đầu tiên.



**Câu 4:** Sinh viên được cung cấp hàm sắp xếp như sau, yêu cầu các bạn viết thêm hàm tìm kiếm nhị phân. Hãy viết hàm sắp xếp sau vào tập tin `caidat_lab1.cpp`.

Mã lệnh
<pre>void sapXep(int a[], int n){     for (int i = 0; i &lt; n - 1; i++)         for (int j = i + 1; j &lt; n; j++)             if (a[j] &lt; a[i]){                 int tam = a[i];                 a[i] = a[j];                 a[j] = tam;             } }</pre>

**Yêu cầu:**

- So sánh sự khác nhau giữa hàm tìm kiếm tuần tự và tìm kiếm nhị phân.
- Vào tập tin `chuongtrinh_lab1.cpp` sử dụng hàm tìm kiếm nhị phân, nhập vào số x cần tìm. Chương trình xuất ra tìm thấy x có trong mảng hay không?

**Câu 5:** Sinh viên hoàn thành các bài tập sau:

- Viết hàm tính tổng các phần tử âm có trong mảng.
- Viết hàm đếm các số là số nguyên tố
- Viết hàm xuất ra các số hoàn thiện có trong mảng (số hoàn thiện là số mà có tổng tất cả các ước (trừ nó) bằng chính nó). Ví dụ: 6 có các ước số là 1,2,3 và  $1 + 2 + 3 = 6$  nên 6 là số hoàn thiện.

#### IV. BÀI TẬP VỀ NHÀ

**Câu 1:** Đếm số lần xuất hiện của giá trị X có trong mảng.

**Câu 2:** Tìm vị trí xuất hiện của phần tử X có trong mảng A. Thay thế giá trị X đó thành giá trị Y.

**Câu 3:** Kiểm tra xem mảng có sắp xếp tăng dần hay không?

**Câu 4:** Viết hàm chèn phần tử X, vào vị trí k có trong mảng.

## BÀI 2: DANH SÁCH ĐẶC – GIẢI THUẬT TÌM KIẾM

### I. MỤC TIÊU:

Sau khi thực hành xong bài này, sinh viên phải biết:

- Tạo mới một danh sách.
- Thêm một phần tử vào danh sách.
- Tìm kiếm phần tử trong danh sách.
- Loại bỏ một phần tử ra khỏi danh sách.
- Cập nhật (sửa đổi) giá trị cho một phần tử trong danh sách.
- Ghép nhiều danh sách thành 1 danh sách.
- Tìm kiếm tuần tự, tìm kiếm nhị phân

### II. TÓM TẮT LÝ THUYẾT:

Nội dung	Ví dụ
<b>1. Danh Sách Đặc:</b>	
<b>Biểu diễn danh sách đặc:</b>  Giả sử chúng ta quy ước chiều dài tối đa của danh sách đặc là 50, khi đó cấu trúc dữ liệu để biểu diễn danh sách đặc như ví dụ bên:	<code>const int MAX=50;</code>  Hoặc:  <code>#define MAX 50</code>
<b>Cài đặt danh sách đặc:</b>  <u>Khai báo tổng quát:</u>  <code>struct &lt;ten_cau_truc&gt;{      int n;//so phan tu trong ds      &lt;kiểu    dữ    liệu&gt;     &lt;ten_mang&gt;&lt;[số    phần</code>	<code>const int MAX=50;  struct DaySo {      int ds[MAX];      int n;  };</code>

<pre>tử]&gt;;  };</pre>	
<p><b>Khởi tạo danh sách đặc:</b></p> <p>Đặt số phần tử trong danh sách bằng 0</p>	<pre>void khoiTao (DaySo &amp;x){     x.n=0; }</pre>
<p><b>Kiểm tra danh sách có rỗng hay không:</b></p> <p>// Nếu số phần tử bằng không trả về 1; ngược lại trả về 0</p>	<pre>int isEmpty (DaySo x){     return (x.n==0?1:0); }</pre>
<p><b>Kiểm tra danh sách có đầy hay không:</b></p> <p>// Nếu số lượng phần tử có trong danh sách bằng MAX trả về 1; ngược lại trả về 0</p>	<pre>int isFull (DaySo x){     return (x.n==MAX?1:0); }</pre>
<p><b>Xác định số phần tử của danh sách:</b></p> <p>//Trả về số phần tử của danh sách</p>	<pre>int listSize (DaySo x){     return x.n; }</pre>
<p><b>Truy xuất giá trị của phần tử tại vị trí pos trong danh sách đặc:</b></p> <p>// Sinh viên tham khảo ví</p>	<pre>int truyXuat_ViTri(DaySo x,int pos){     if(pos&lt;0    pos&gt;listSize(x)-1)         cout &lt;&lt;"Vi tri "&lt;&lt;pos&lt;&lt;" không         hop le";     else         if(isEmpty(x))</pre>

dụ	<pre>        cout&lt;&lt;"DS bi rong";         else return x.ds[pos];     } }</pre>
<b>Thêm một phần tử vào danh sách đặc tại vị trí pos cho trước</b>  // Sinh viên tham khảo ví dụ	<pre>void chen_ViTri(DaySo &amp;x, int pos, int a){     if (pos&lt;0    pos&gt;listsize(x))         cout &lt;&lt; "\n Vi tri " &lt;&lt; pos         &lt;&lt; " khong hop le";     else         if (isEmpty(x) == 1){             if (pos == 0){                 x.ds[0] = a;                 x.n++;             }             else                 cout &lt;&lt; "\n Vi tri khong                 hop le";         }         else if (isFull(x) == 1)             cout &lt;&lt; "\n Danh sach             day";         else{             for (int i = listSize(x) - 1; i &gt;= pos; i-             -)                 x.ds[i + 1] = x.ds[i];             x.ds[pos] = a;             x.n++;         }     } }</pre>
<b>Xóa một phần tử ra khỏi danh sách đặc tại vị trí pos:</b>  // Sinh viên tham khảo ví dụ	<pre>void xoa_ViTri(DaySo &amp;x, int pos){     int i;     if(pos&lt;0    pos&gt;listSize(x)-1)         cout &lt;&lt;"Vi tri"&lt;&lt;pos&lt;&lt;"khong hop le         !";     else         if(isEmpty(x)) cout&lt;&lt;"DS bi rong";         else {             for(i=pos+1;i&lt;=listSize(x);i++)                 x.ds[i-1]= x.ds[i];         } }</pre>

	<pre>        x.n--;     } }</pre>
<p><b>Thay thế một phần tử trong danh sách đặc bởi một phần tử khác tại vị trí pos:</b></p> <p>// Sinh viên tham khảo ví dụ</p>	<pre>void thayThe_ViTri (DaySo &amp;x, int pos, int a){     if (pos&lt;0  pos&gt;=listSize(x))         cout&lt;&lt;"Vi tri "&lt;&lt;pos&lt;&lt;"khong tim thay !";     else if (isEmpty(x))         cout&lt;&lt;"Danh sach bi rong " ;     else x.ds[pos]=a ; }</pre>
<p><b>2. Tìm kiếm:</b> Tùy theo dữ liệu đã được sắp xếp hay chưa mà ta có 2 phương pháp tìm kiếm: tuyến tính (tuần tự) và nhị phân. Dưới đây là đoạn mã cài đặt giải thuật tìm kiếm tuần tự và tìm kiếm nhị phân.</p>	
<p><b>a. Tìm kiếm tuần tự</b></p>	<pre>int timKiem_TuanTu(DaySo x,int a) {     int i=0;     while(i&lt;x.n &amp;&amp; x.ds[i]!=a)         i++;     if (i==x.n)         return -1 ;//Khong tim thay     return i; }</pre>

**b. Tìm kiếm nhị phân:**

```
//Tìm kiếm nhị phân
Int timKiem_NhiPhan(DaySo x, int a)
{
    int bottom=0, top=x.n-1; int mid;

    do{
        mid=(bottom+top)/2;
        if(a == x.ds[mid]) return
mid;
        else
            if (a<x.ds[mid])
top=mid-1;
            else bottom= mid+1;
    } while (bottom<=top);
    return -1;
}
```

**III. NỘI DUNG THỰC HÀNH**

**Câu 1:** Cho đoạn mã sau, yêu cầu sinh viết và chạy chương trình.

- File **thuvien\_lab2.h** như sau:

**Mã lệnh**

```
#include<iostream>
using namespace std;
const int MAX = 50;
//Cau truc DS DAC
struct DaySo{
    int ds[MAX];
    int n;           //So Phan tu ds dac
};
void khoiTao(DaySo &x);
int isEmpty(DaySo x);
int isFull(DaySo x);
int listSize(DaySo x);
void doc_DanhSach(DaySo &x);
void xuat_DanhSach(DaySo x);
```

- File `caidat_lab2.cpp` như sau:

**Mã lệnh**

```
#include "thuvien_lab2.h"
//Hàm khởi tạo
void khoiTao(DaySo &x){
    x.n = 0;
}
//Hàm kiểm tra rỗng
int isEmpty(DaySo x){
    return (x.n == 0 ? 1 : 0);
}
//Hàm kiểm tra đầy
int isFull(DaySo x){
    return (x.n == MAX ? 1 : 0);
}
//Hàm trả về số nút của DS đặc
int listSize(DaySo x){
    return x.n;
}
/*****Hàm nhập day so vào ds DAC*****/
void doc_DanhSach(DaySo &x){
    cout << "\n Nhập vào số phần tử của DS:";
    cin >> x.n;
    for (int i = 0; i < x.n; i++){
        cout << "\n Nhập vào phần tử thứ " << i + 1 <<
        ":";
        cin >> x.ds[i];
    }
}
/*****Hàm xuất ra DS DAC*****/
void xuat_DanhSach(DaySo x){
    cout << "\n Phần tử trong DS:";
    for (int i = 0; i < x.n; i++)
        cout << x.ds[i] << " ";
}
```

**Yêu cầu:**

- a/. Viết vào file **chuongtrinh\_lab2.cpp** để gọi và chạy các hàm đã viết.
- b/. Cho biết ý nghĩa của đoạn chương trình trên.

**Câu 2:** Sau đây là đoạn mã thêm một phần tử vào DS đặc ở vị trí bất kỳ, lời gọi hàm trong thân hàm, sinh viên hãy soạn thảo lại mã nguồn và chạy thử chương trình.

- File **caidat\_lab2.cpp** sinh viên soạn đoạn mã sau vào:

**Mã lệnh**

```
void chen_ViTri(DaySo &x, int pos, int a){
    if (pos<0 || pos>listSize(x))
        cout << "\n Vi tri " << pos << " khong hop le";
    else
        if (isEmpty(x) == 1){
            if (pos == 0){
                x.ds[0] = a;
                x.n++;
            }
            else
                cout << "\n Vi tri khong hop le";
        }
        else if (isFull(x) == 1)
            cout << "\n Danh sach day";
        else{
            for (int i = listSize(x) - 1; i >= pos; i--)
                x.ds[i + 1] = x.ds[i];
            x.ds[pos] = a;
            x.n++;
        }
}
```

- ở file **chuongtrinh\_lab2.cpp** như sau:
-



Mã lệnh

```
#include "thuvien_lab2.h"
void main(){
    DaySo x;
    int pos, a, k;
    khoiTao(x);
    doc_DanhSach(x);
    xuat_DanhSach(x);
    cout << "\n Nhập phần tử cần chen:";
    cin >> k;
    cout << "\n Nhập vị trí cần chen:";
    cin >> pos;
    chen_ViTri(x, pos, k);
    xuat_DanhSach(x);
}
```

**Câu 3:** Cho đoạn mã xóa phần tử ở vị trí pos, yêu cầu sinh viên viết lại và gọi hàm trong void main ().

Mã lệnh

```
void xoa_ViTri(DaySo &x, int pos){
    int i;
    if (pos<0 || pos>listSize(x) - 1)
        cout << "Vị trí" << pos << "không hợp lệ !";
    else
        if (isEmpty(x)) cout << "DS bị rỗng";
        else{
            for (i = pos + 1; i <= listSize(x); i++)
                x.ds[i - 1] = x.ds[i];
            x.n--;
        }
}
```

**Câu 4:** Sau đây là hàm xóa một phần tử có giá trị bằng k, có trong DS Đặc. Nhưng hàm **đang có lỗi sai**, yêu cầu sinh viên sửa lại hàm cho đúng. Viết void main () gọi hàm trên.

**Mã lệnh**

```
void xoa_GiaTri_K(DaySo &x, int k){
    for (int i = 0; i<x.n; i++){
        if (x.ds[i] == k){
            for (int j = i; j< x.n - 1; j++)
                x.ds[j] = x.ds[j + 1];
            x.n--;
        }
    }
}
```

**Câu 5:** Đoạn mã dưới đây là hàm tìm kiếm tuần tự, cách gọi hàm tìm kiếm tuần tự trong **void main**.

**Yêu cầu:**

- Sinh viên soạn thảo và chạy chương trình.
- Sửa lại hàm main, để xuất ra vị trí tìm thấy phần tử a cần tìm ở vị trí thứ mấy có trong DS. Ngược lại thì xuất ra không tìm thấy.

**Mã lệnh**

```
int timKiem_TuanTu(DaySo x, int a){
    int i = 0;
    while (i<x.n && x.ds[i] != a)
        i++;
    if (i == x.n)
        return -1;//Khong tim thay
    return i;
}
```

- File **chuongtrinh\_lab2.cpp** như sau:

**Mã lệnh**

```
#include"thuvien_lab2.h"
void main(){
    DaySo x;
    int pos, a, k;
    khoiTao(x);
    doc_DanhSach(x);
    xuat_DanhSach(x);
}
```

```
cout << "\n Nhap phan tu k can tim:";
cin >> k;
if (timKiem_TuanTu(x, k) == -1)
    cout << "\n Khong tim thay";
else
    cout << "\n Tim thay";
}
```

**Câu 6:** Sau đây là hàm sắp xếp dãy số tăng dần, yêu cầu các bạn viết hàm tìm kiếm nhị phân và gọi lại hàm trong main.

Mã lệnh
<pre>void sapXep(DaySo &amp;x){     for (int i = 0; i &lt; x.n - 1; i++)         for (int j = i + 1; j &lt; x.n; j++)             if (x.ds[j] &lt; x.ds[i]){                 int tam = x.ds[i];                 x.ds[i] = x.ds[j];                 x.ds[j] = tam;             } }</pre>

**Câu 7:** Sinh viên tự viết chương trình để thực hiện các công việc sau:

- Nhập/xuất danh sách có  $n$  phần tử là số nguyên.
- Xuất số lượng phần tử có giá trị là  $x$  trong mảng ( $x$  là số nhập vào khi chạy chương trình).
- Xuất lần lượt tất cả vị trí của phần tử có giá trị là  $x$  trong mảng. ( $x$  là phần tử được nhập vào)

#### **IV. BÀI TẬP VỀ NHÀ**

**Câu 1:** Sinh viên tự viết chương trình sau:

- Ghép hai danh sách đặc lại với nhau thành 1 danh sách kết quả. Giá trị các phần tử trong danh sách kết quả có nội dung là giá trị các phần tử của danh sách 1 rồi đến danh sách 2.
- Tách danh sách ban đầu thành 2 danh sách: một danh sách chứa phần tử chẵn, một danh sách chứa phần tử lẻ

## BÀI 3: DANH SÁCH ĐẶC – TRÊN ĐỐI TƯỢNG

### I. MỤC TIÊU:

Sau khi thực hành xong bài này, sinh viên phải:

- Tạo được cấu trúc của một đối tượng
- Tìm kiếm trên đối tượng
- Thêm, xóa bỏ thành phần trong đối tượng

### II. TÓM TẮT LÝ THUYẾT

Nội dung	Ví dụ
<b>1. Cấu trúc</b>	
- Khai báo cấu trúc	<code>struct &lt;tên kiểu cấu trúc&gt;{ &lt;kiểu dữ liệu 1&gt; tên thành phần 1; ... &lt;kiểu dữ liệu n&gt; tên thành phần n; };</code>
- Một số lưu ý:	<ul style="list-style-type: none"> <li>- Bắt buộc phải có <b>;</b> sau dấu }</li> <li>- Tên của kiểu struct <b>nên</b> bắt đầu bằng ký tự hoa.</li> <li>- Các thành phần thuộc tính cùng kiểu dữ liệu nên cách nhau dấu , như các điểm toán, lý, hóa.</li> </ul>
- Truy xuất biến kiểu cấu trúc	<ul style="list-style-type: none"> <li>- <b>Đối với thành phần là 1 kiểu dữ liệu cơ bản:</b>     &lt;tên biến cấu trúc&gt;.&lt;tên thành phần&gt;</li> <li>- <b>Đối với thành phần là 1 kiểu cấu trúc:</b>     &lt;tên biến cấu trúc 1&gt;.&lt;tên biến cấu trúc 2&gt;.&lt;tên thành phần&gt;</li> </ul>

<p>- Ví dụ khai báo cấu trúc sinh viên gồm có:</p> <pre>masv,      hotensv, namsinh,    toan, ly,hoa, dtb:</pre>	<pre>struct SinhVien{     char masv[10];     char hotensv[30],     float toan,ly,hoa,dtb; };</pre>
--	--

### III. NỘI DUNG THỰC HÀNH

**Câu 1:** Đoạn mã sau dùng để khai báo cấu trúc một môn học, có hàm nhập, xuất ra một môn học:

- Trong file **thuvien\_lab3.h** có đoạn mã như sau:

Mã lệnh
<pre>#include&lt;iostream&gt; #include&lt;string.h&gt; using namespace std; const int MAX = 50; struct MonHoc{     char mamh[10];     char tenmh[30];     int sotc; }; void nhapMonHoc(MonHoc &amp;mh); void xuatMonHoc(MonHoc mh);</pre>

- Trong file **caidat\_lab3.cpp**, đoạn mã như sau:

Mã lệnh
<pre>#include"thuvien_lab3.h" void nhapMonHoc(MonHoc &amp;mh){     cout &lt;&lt; "\n NHap ma mh:";     cin &gt;&gt; mh.mamh;     cout &lt;&lt; "\n NHap Ten MH:";     cin.ignore();</pre>

```
cin.getline(mh.tenmh, 30);
cout << "\n Nhập so tin chi:";
cin >> mh.sotc;
}
void xuấtMonHoc(MonHoc mh){
    cout << "Ma MH" << "\t" << "TenMH" << "\t\t" << "So
TC" << endl;
    cout << mh.mamh << "\t" << mh.tenmh << "\t\t" <<
mh.sotc<<endl;
}
```

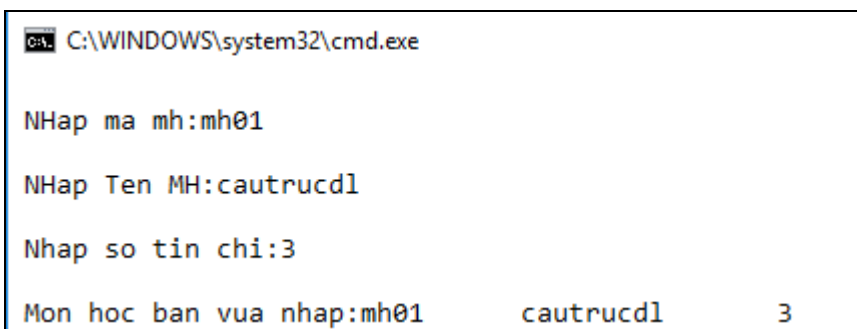
- Trong file **chuongtrinh\_lab3.cpp**, đoạn mã như sau:

**Mã lệnh**

```
#include"thuvien_lab3.h"
void main(){
    MonHoc mh;
    DanhSachMonHoc ds;
    nhapMonHoc(mh);
    cout << "\n Mon hoc ban vua nhap:" << endl;
    xuấtMonHoc(mh);
}
```

**Yêu cầu:**

- Sinh viên nhập đoạn mã trên và chạy chương trình xuất ra kết quả như sau:



```
C:\WINDOWS\system32\cmd.exe

NHap ma mh: mh01

NHap Ten MH: cautrucdl

NHap so tin chi: 3

Mon hoc ban vua nhap: mh01 cautrucdl 3
```

**Câu 2:** Sau đây là cấu trúc của một danh sách môn học.

- Trong file **thuvien\_lab3.h** có đoạn mã như sau:

Mã lệnh

```
#include<iostream>
#include<string.h>
using namespace std;
const int MAX = 50;
struct MonHoc{
    char mamh[10];
    char tenmh[30];
    int sotc;
};
struct DanhSachMonHoc{
    int so;
    MonHoc data[MAX];
};
void nhapMonHoc(MonHoc &mh);
void xuatMonHoc(MonHoc mh);
void nhapDanhSachMonHoc(DanhSachMonHoc &ds);
void xuatDanhSachMonHoc(DanhSachMonHoc ds);
```

- Trong file **caidat\_lab3.cpp**, đoạn mã như sau:

Mã lệnh

```
void nhapDanhSachMonHoc(DanhSachMonHoc &ds){
    cout << "\n Nhập số lượng môn học:";
    cin >> ds.so;
    cout << "\n Nhập thông tin từng môn học:";
    for (int i = 0; i<ds.so; i++){
        cout << "\n Thông tin Môn Học Thu:" << i + 1 <<
endl;
        nhapMonHoc(ds.data[i]);
    }
}
void xuatDanhSachMonHoc(DanhSachMonHoc ds){
    cout << "\n Danh sách các môn học:" << endl;
    cout << "\n MaMH \t TenMH \t so tc"<<endl;
    for (int i = 0; i<ds.so; i++)
        xuatMonHoc(ds.data[i]);
}
```

- Trong file **chuongtrinh\_lab3.cpp**, đoạn mã như sau:

Mã lệnh

```
#include"thuvien_lab3.h"
void main(){
    MonHoc mh;
    DanhSachMonHoc ds;
    nhapDanhSachMonHoc(ds);
    xuatDanhSachMonHoc(ds);
}
```

**Yêu cầu:**

- Sinh viên bổ sung vào các hàm **nhapMH** để xuất ra một môn học, **XuatMH** để xuất một môn học để chương trình trên có thể chạy được.
- Các bạn hãy nhập vào danh sách 2 môn học để chạy thử.

**Câu 3:** Sau đây là hàm thêm vào môn học (**them\_MonHoc**) vào vị trí (**vt**) bất kỳ trong danh sách môn học (**DanhSachMonHoc**) đã có như sau:

Mã lệnh

```
void them_MonHoc_ViTri(DanhSachMonHoc &ds, int vt, MonHoc mh){
    if (vt<0 || vt>ds.so)
        cout << "\n Vi tri khong hop le";
    else{
        for (int i = ds.so - 1; i >= vt; i--)
            ds.data[i + 1] = ds.data[i];
        ds.data[vt] = mh;
        ds.so++;
    }
}
```

**Yêu cầu:**

- Sinh viên bổ sung vào những đoạn mã đang còn thiếu



- Viết trong hàm void main để chạy được chương trình, nhằm thêm một môn học mới vào vị trí bất kỳ có trong **DanhSachMonHoc** đã có.

**Câu 4:** Dưới đây là đoạn mã xóa một môn học (**xoa\_MonHoc\_ViTri**) tại vị trí (**vt**) bất kỳ có trong danh sách môn học (**DanhSachMonHoc**):

Mã lệnh
<pre>void xoa_MonHoc_ViTri(DanhSachMonHoc &amp;ds, int vt){     if (vt&lt;0    vt&gt;ds.so)         cout &lt;&lt; "\n vi tri khong hop le";     else{         for (int i = vt + 1; i &lt;= ds.so; i++)             ds.data[i - 1] = ds.data[i];         ds.so--;     } }</pre>

**Yêu cầu:**

- Viết **void main** gọi hàm trên
- Áp dụng tương tự, các bạn viết hàm xóa 1 môn học (**xoa\_maMonHoc**) trong danh sách môn học đang có (**DanhSachMonHoc**) khi nhập vào **mã môn học** theo nguyên mẫu hàm sau: **void xoa\_maMonHoc (DanhSachMonHoc &ds, char \*mamh) .**

**Câu 5:** Dưới đây là đoạn mã của hàm tìm kiếm, nhập vào mã môn học, sử dụng cấu trúc tìm kiếm tuần tự áp dụng trên cấu trúc danh sách môn học để tìm ra môn học thỏa mãn điều kiện nhập vào.

- Trong file **caidat\_lab3.cpp**, đoạn mã như sau:

Mã lệnh
<pre>int timKiem_TuanTu(DanhSachMonHoc ds, char* ma){     int i = 0;     while (i&lt;ds.so &amp;&amp; strcmp(ds.data[i].mamh, ma) != 0)         i++; }</pre>

```

        if (i == ds.so)
            return -1; //Không tìm thấy
        return i;
    }

```

- Trong file **chuongtrinh\_lab3.cpp**, đoạn mã như sau:

Mã lệnh
<pre> #include "thuvien_lab3.h" void main(){     MonHoc mh;     DanhSachMonHoc ds;     nhapDanhSachMonHoc(ds);     xuatDanhSachMonHoc(ds);     char mamh[10];     cout &lt;&lt; "\n Nhập mã MH muốn tìm:";     cin &gt;&gt; mh.mamh;     int k = timKiem_TuanTu(ds, mh.mamh);     if (k == -1)         cout &lt;&lt; "\n Không có mã mh trong DS";     else{         cout &lt;&lt; "\n Có môn học bạn cần tìm:";         xuatMonHoc(ds.data[k]);     } } </pre>

### Yêu cầu:

- Nhập đoạn mã trên và chạy chương trình.
- Giải thích lệnh **strcmp(ds.data[i].mamh, ma)** ở đoạn mã trên

**Câu 6:** Sau đây là hàm sắp xếp **selectionSort** để sắp xếp một **DanhSachMonHoc** đã có theo số tín chỉ tăng dần.

Mã lệnh
<pre> void selectionSort(DanhSachMonHoc &amp;ds){     MonHoc a;     int min;     for (int i = 0; i &lt; ds.so - 1; i++){ </pre>

```

        min = i;
        for (int j = i + 1; j < ds.so; j++)
            if (ds.data[j].sotc < ds.data[min].sotc)
                min = j;
        a = ds.data[i];
        ds.data[i] = ds.data[min];
        ds.data[min] = a;
    }
}

```

**Yêu cầu:**

- a/. Viết hàm **selectionSort** vào chương trình và viết void main gọi hàm trên
- b/. Viết hàm **insertionSort** để sắp xếp môn học giảm dần theo tín chỉ.

#### IV. BÀI TẬP VỀ NHÀ

**Câu 1:** Cho cấu trúc của một sinh viên như sau:

Mã lệnh
<pre> struct SinhVien{     char masv[10];     char tensv[30];     float toan, ly, hoa; }; struct DanhSachSinhVien{     int sosv;     SinhVien data[MAX]; }; </pre>

**Yêu cầu:**

- a/. Viết các hàm thêm một sinh viên tại vị trí bất kỳ.
- b/. Viết hàm xóa sinh viên tại vị trí bất kỳ.
- c/. Viết hàm tìm kiếm, tìm sinh viên theo mã sinh viên,.
- d/. viết hàm sắp xếp danh sách sinh viên giảm dần theo điểm trung bình (**đtb**) tính theo công thức **đtb = (toán\*2+ly+hoa\*3)/6**.

## BÀI 4: DANH SÁCH LIÊN KẾT ĐƠN

### I. MỤC TIÊU

Sau khi thực hành xong bài này, sinh viên phải biết cách:

- Khai báo một DSLK
- Thêm vào một phần tử trong DSLK
- Tìm một phần tử trong DSLK
- Xóa phần tử trong DSLK

### II. TÓM TẮT LÝ THUYẾT

Nội dung	Ví dụ
<b>1. Danh sách liên kết đơn:</b> <ul style="list-style-type: none"> <li>• <b>Định nghĩa :</b> DSLK là danh sách mà các phần tử được nối kết với nhau nhờ vào vùng liên kết của chúng.</li> <li>• Danh sách liên kết đơn là danh sách liên kết trong đó mỗi phần tử chỉ có một liên kết với phần tử tiếp theo.</li> <li>• <b>Cách biểu diễn:</b> Mỗi phần tử của danh sách liên kết đơn gồm hai vùng: <ul style="list-style-type: none"> <li>- Vùng chứa thông tin <u>data</u></li> <li>- Vùng chứa liên kết đến phần tử kế <u>link</u></li> </ul> </li> <li>• <b>Hình ảnh minh họa:</b></li> </ul>	
- Khai báo kiểu dữ liệu cho danh sách số nguyên:	<pre>typedef int TYPEINFO; struct Node{     TYPEINFO data;     Node* link; }; typedef Node* Nodeptr; typedef Nodeptr list;</pre>
- Khởi tạo danh sách liên kết	<pre>void khoiTao(Nodeptr &amp;list){     list =NULL; } </pre>
- Hàm tạo một node:	<pre>Nodeptr tao_Node(TYPEINFO x){     Nodeptr p;</pre>

	<pre> p=new Node; p-&gt;data =x; p-&gt;link=NULL; return p; } </pre>
- Hàm giải phóng bộ nhớ	<pre> void giaiPhong(Nodeptr &amp;list){ Nodeptr p=list; while(p!=NULL){ list=list-&gt;link; delete p; p=list; } } </pre>
- Kiểm tra danh sách rỗng hay không	<pre> int isEmpty(Nodeptr list){ return(list==NULL ? 1:0); } </pre>
- Thêm 1 phần tử vào đầu DSLK đơn:	<pre> Nodeptr them_Dau(Nodeptr &amp;list,TYPEINFO x){ Nodeptr p; p=tao_Node(x); p-&gt;link=list; list=p; return p; } </pre>
- Thêm 1 phần tử vào giữa DSLK đơn (Thêm sau phần tử q):	<pre> NODEPTR Chensau_q(Nodeptr q, TYPEINFO x){ Nodeptr p=tao_Node(x); p-&gt;link = q-&gt;link; q-&gt;link=p; return q; } </pre>

<p>- Thêm 1 phần tử vào cuối DSLK đơn:</p>	<pre>//Ham Them CUOI Nodeptr them_Cuoi(Nodeptr &amp;list, TYPEINFO x){     Nodeptr p;     p=tao_Node(x);     if(list==NULL)         list=p;     else{         Nodeptr q=list;         while(q-&gt;link!=NULL)             q = q-&gt;link;         q-&gt;link=p;     }     return p; }</pre>
<p>- Hàm xóa 1 phần tử ở đầu DSLK đơn</p>	<pre>void xoa_Dau(Nodeptr &amp;list){     Nodeptr p;     if(list!=NULL){         p=list;         list=p-&gt;link;         delete p;     } }</pre>
<p>- Hàm xóa 1 phần tử có giá trị là x chứa trong DSLK đơn</p>	<pre>void xoa_Node_X(Nodeptr &amp;list, int x){     Nodeptr p, before;     if(list-&gt;data ==x)         xoa_Dau(list);     else{         p=list;         before =list;         while((p!=NULL) &amp;&amp; (p-&gt;data!=x)){             before =p;             p=p-&gt;link;         }         if(p!=NULL){             before-&gt;link=p-&gt;link;             delete p;         }     } }</pre>

	<pre>         }     } } </pre>
- Hàm xóa 1 phần tử ở cuối trong DSLK đơn	<pre> void xoa_Cuoi(Nodeptr &amp;list) {     Nodeptr p, before;     p=list;     before=list;     if(p-&gt;link==NULL){         delete p;         list=NULL;     }     else{         while(p-&gt;link!=NULL){             before=p;             p=p-&gt;link;         }         if(p-&gt;link==NULL){             delete p;             before-             &gt;link=NULL;         }     } } </pre>

### III. NỘI DUNG THỰC HÀNH

**Câu 1:** Dưới đây là đoạn mã nguồn dùng để nhập dữ liệu cho một danh sách liên kết.

- Trong file **thuvien\_lab4.h**, đoạn mã như sau:

Mã lệnh
<pre> #include&lt;iostream&gt; using namespace std; typedef int TYPEINFO;  struct Node{ </pre>

```

        TYPEINFO data;
        Node *link;
};
typedef Node *Nodeptr;
void khoiTao(Nodeptr &list);
int isEmpty(Nodeptr list);
Nodeptr tao_Node(TYPEINFO x);
Nodeptr them_Dau(Nodeptr &list, TYPEINFO x);
void nhap_DanhSach(Nodeptr &list);
void xuat_DanhSach(Nodeptr list);
void giaiPhong(Nodeptr &list);

```

- Trong file **caidat\_lab4.cpp**, đoạn mã như sau:

#### Mã lệnh

```

#include"thuvien_lab4.h"
void khoiTao(Nodeptr &list){
    list = NULL;
}
int isEmpty(Nodeptr list){
    return (list == NULL ? 1 : 0);
}
Nodeptr tao_Node(TYPEINFO x){
    Nodeptr p;
    p = new Node;
    p->data = x;
    p->link = NULL;
    return p;
}
Nodeptr them_Dau(Nodeptr &list, TYPEINFO x){
    Nodeptr p;
    p = tao_Node(x);
    p->link = list;
    list = p;
    return p;
}
void nhap_DanhSach(Nodeptr &list)
{
    khoiTao(list);
    int x;
    do{

```



```

        cout << "\n Nhap vao x= (thoat -99)";
        cin >> x;
        if (x == -99)
            break;
        them_Dau(list, x); //Goi ham them ??U
    } while (1);
}
void xuat_DanhSach(Nodeptr list){
    Nodeptr p = list;
    while (p != NULL){
        cout << p->data << " ";
        p = p->link;
    }
}
void giaiPhong(Nodeptr &list){
    Nodeptr p = list;
    while (p != NULL){
        list = list->link;
        delete p;
        p = list;
    }
}
}

```

- Trong file **chuongtrinh\_lab4.cpp**, đoạn mã như sau:

#### Mã lệnh

```

#include"thuvien_lab4.h"
void main(){
    Nodeptr list;
    khoiTao(list);
    nhap_DanhSach(list);
    cout << "\n DSLK vua nhap:";
    xuat_DanhSach(list);
    giaiPhong(list);
    system("pause");
}

```

#### Yêu cầu:

- Sinh viên biên dịch và cho biết danh sách được tạo sử dụng kỹ thuật chèn đầu hay chèn cuối.

b. Cho biết ý nghĩa hàm **void giaiPhong(Nodeptr &list)**.

c. Ghi kết quả xuất ra khi nhập vào dãy số sau: **10, 8, 7, 9, 20**.

**Câu 2:** Sinh viên viết lại hàm nhập dữ liệu cho danh sách liên kết sử dụng kỹ thuật chèn cuối.

**Câu 3:** Sau đây là đoạn mã tìm kiếm phần tử có giá trị x có trong danh sách liên kết. Nếu tìm thấy trả về node đấy, nếu không trả về NULL.

- Trong file **caidat\_lab4.cpp** như sau:

Mã lệnh
<pre>Nodeptr tim_Node_X(Nodeptr list, TYPEINFO x){     Nodeptr p = list;     while (p != NULL){         if (p-&gt;data == x)             return p;         p = p-&gt;link;     }     return p; }</pre>

**Yêu cầu:**

- Sinh viên hãy gọi hàm vừa viết ở trên trong **void main**.

**Câu 4:** Đoạn mã sau là chèn node có dữ liệu là x vào sau node do con trỏ q quản lý, đoạn mã như sau:

Mã lệnh
<pre>Nodeptr chenSau_Q(Nodeptr q, TYPEINFO x) {     Nodeptr p =tao_Node(x);     p-&gt;link = q-&gt;link;     q-&gt;link = p;     return q; }</pre>

**Yêu cầu:**

- Viết các hàm để hoàn thiện chèn phần tử có nội dung là x được thêm vào sau phần tử do con trỏ q quản lý có trong danh sách liên kết.

**Câu 5:** Sau đây là đoạn mã xóa đầu, sinh viên nhập và chạy chương trình.

- Trong file **caidat\_lab4.cpp** như sau:

Mã lệnh
<pre>void xoa_Dau(Nodeptr &amp;list) {     Nodeptr p;     if (list != NULL){         p = list;         list = p-&gt;link;         delete p;     } }</pre>

- Trong file **chuongtrinh\_lab4.cpp** như sau:

Mã lệnh
<pre>#include "thuvien_lab4.h" void main(){     Nodeptr list;     khoiTao(list);     nhap_DanhSach(list);     cout &lt;&lt; "\n DSLK vua nhap:";     xuat_DanhSach(list);     xoa_Dau(list);     cout &lt;&lt; "\n DSLK sau khi xoa phan tu dau:";     xuat_DanhSach(list);     giaiPhong(list);     system("pause"); }</pre>

**Yêu cầu:**

- Viết hàm xóa node cuối của danh sách liên kết và chạy chương trình.

**Câu 6:** Viết hàm xóa node có nội dung là x có trong DSLK, biết nguyên mẫu hàm như sau: `void del_node_x(NODEPTR &list, int x)`

**Ví dụ:** có danh sách liên kết gồm các node sau: 5, 8, -12, 9, 18

Phần tử x cần xóa là: -12.

Xuất ra danh sách liên kết sau khi xóa node x là: 5, 8, 9, 18

**Câu 7:** Viết hàm đảo ngược một DSLK.

**Ví dụ:** danh sách liên kết ban đầu: **9 -10 25 11 -9**

- Sau khi đảo ngược ta được danh sách: **-9, 11, 25, -10, 9**

#### **IV. BÀI TẬP VỀ NHÀ**

**Câu 1:** Viết hàm tách một DSLK ra thành 2 DS, một DSLK chứa phần tử chẵn, một DSLK chứa phần tử lẻ.

Ví dụ: Danh sách ban đầu: 9, -8, 7, 11, 20, -2, 15

Xây dựng hàm tách danh sách liên kết ta được:

- **Danh sách 1 chứa phần tử là các số chẵn:** -8      20      -2
- **Danh sách 2 chứa phần tử là các số lẻ:** 9      7      11      15

**Câu 2:** Viết hàm xóa tất cả các phần tử có nội dung là x có trong DSLK, biết nguyên mẫu hàm: `void del_All_x(NODEPTR &list, int x)`.

Ví dụ: danh sách ban đầu: **9 11 5 9 2 9 -17**

- Phần tử cần xóa **x = 9**
- Danh sách còn lại sau khi xóa tất cả các phần tử 9:

**11 5 2 -17**

## **BÀI 5: DANH SÁCH LIÊN KẾT ĐƠN – TRÊN ĐỐI TƯỢNG**

### **I. MỤC TIÊU:**

- Sinh viên sẽ biết cách cài đặt DSLK đơn cho các đối tượng cấu trúc
- Xây dựng các phép toán cơ bản trên đối tượng cấu trúc cài đặt bằng DSLK đơn.

### **II. TÓM TẮT LÝ THUYẾT**

<b>Nội dung</b>	<b>Ví dụ</b>
<b>1. Cấu trúc</b>	
- Khai báo cấu trúc	<pre><b>struct</b> &lt;tên kiểu cấu trúc&gt;{     &lt;kiểu dữ liệu 1&gt; tên thành phần 1;     ...     &lt;kiểu dữ liệu n&gt; tên thành phần n; };</pre>
- Một số lưu ý:	<ul style="list-style-type: none"><li>- Bắt buộc phải có ; sau dấu }</li><li>- Tên của kiểu struct nên bắt đầu bằng ký tự hoa.</li><li>- Các thành phần thuộc tính cùng kiểu dữ liệu nên cách nhau dấu , như các điểm toán, lý, hóa.</li></ul>
- Truy xuất biến kiểu cấu trúc	<ul style="list-style-type: none"><li>- Đối với thành phần là 1 kiểu dữ liệu cơ bản:      &lt;tên biến cấu trúc&gt;.&lt;tên thành phần&gt;</li><li>- Đối với thành phần là 1 kiểu cấu trúc:      &lt;tên biến cấu trúc 1&gt;.&lt;tên biến cấu trúc 2&gt;.&lt;tên thành phần&gt;</li></ul>

<ul style="list-style-type: none"><li>- Ví dụ khai báo cấu trúc sinh viên gồm có các thành phần sau: <code>masv</code>, <code>hotesv</code>, <code>namsinh</code>, <code>toan</code>, <code>ly</code>, <code>hoa</code>, <code>dtb</code>.</li></ul>	<pre>struct SinhVien{     char masv[10];     char tensv[30];     float diem; };</pre>
<ul style="list-style-type: none"><li>- Khai báo cấu trúc danh sách sinh viên</li></ul>	<pre>typedef struct Node* Nodeptr; struct Node{     SinhVien data;     Nodeptr link; };</pre>

### III. NỘI DUNG THỰC HÀNH

**Câu 1:** Đoạn mã sau dùng cấu trúc `SinhVien` gồm có **`masv`**, **`tensv`**, **`diem`** để nhập vào một danh sách sinh viên, và xuất danh sách sinh viên vừa nhập:

- Trong file **`thuvien_lab5.h`** như sau:

Mã lệnh
<pre>#define _CRT_SECURE_NO_WARNINGS #include&lt;iostream&gt; #include&lt;iomanip&gt; #include"string.h" using namespace std; typedef struct Node* Nodeptr; struct SinhVien{     char masv[10];     char tensv[30];     float diem; }; struct Node{     SinhVien data;     Nodeptr link;</pre>

```
};  
void khoiTao(Nodeptr& list);  
int isEmpty(Nodeptr list);  
void giaiPhong(Nodeptr& list);  
Nodeptr tao_Node_SinhVien(SinhVien x);  
Nodeptr them_Dau(Nodeptr& list, SinhVien x);  
bool kiemTra_TrungMa(Nodeptr& list, char* ma);  
void nhap_DSSV(Nodeptr &list);  
void xuat_DSSV(Nodeptr list);
```

- Trong file **caidat\_lab5.cpp** như sau:

**Mã lệnh**

```
#include"thuvien_lab5.h"  
void khoiTao(Nodeptr& list){  
    list = NULL;  
}  
int isEmpty(Nodeptr list){  
    return list == NULL ? 1 : 0;  
}  
void giaiPhong(Nodeptr& list){  
    Nodeptr p = list;  
    while (p != NULL){  
        list = list->link;  
        delete p;  
        p = list;  
    }  
}  
Nodeptr tao_Node_Sv(SinhVien x){  
    Nodeptr p;  
    p = new Node;  
    p->data = x;  
    p->link = NULL;  
    return p;  
}  
//Ham Them 1 sinh vien  
Nodeptr Them_dau(Nodeptr &list, SinhVien x)  
{  
    Nodeptr p = tao_Node_Sv(x);  
    p->link = list;  
    list = p;  
}
```

```
        return p;
    }
    bool kiểmTra_TrungMa(Nodeptr& list, char* ma){
        Nodeptr p = list;
        while (p != NULL){
            if (strcmp(p->data.masv, ma) == 0)
                return true;
            p = p->link;
        }
        return false;
    }
}
//Nhap DSSV
void nhap_DSSV(Nodeptr &list){
    //khởiTạo(list);
    ///char x[10];
    SinhVien sv;
    khởiTạo(list);
    do{
        cout << "\n Nhap MaSV (NHAP 0 DE THOAT):";
        cin.getline(sv.masv, 10);
        while (kiểmTra_TrungMa(list, sv.masv)){
            cout << "\n Ma so sv bi trung";
            cout << "\n Nhap lai ma so sv:";
            cin.getline(sv.masv, 10);
        }
        if (strcmp(sv.masv, "0") == 0) //NHAP MA SO = 0
DE THOAT
            break;
        cout << "\n Nhap Ten SV:";
        cin.getline(sv.tensv, 30);
        cout << "\n NHap Diem SV:";
        cin >> sv.diem;
        cin.ignore();
        Them_dau(list, sv);
    } while (1);
}
void xuất_DSSV(Nodeptr list){
    cout << setw(0) << "MASV" << setw(15) << "TEN SV" <<
    setw(20) << "Diem" << endl;
    while (list != NULL){
        cout << setw(0) << list->data.masv << setw(15)
        << list->data.tensv << setw(20) << list->data.diem << endl;
```



## Bài 5: DANH SÁCH LIÊN KẾT ĐƠN – TRÊN ĐỐI TƯỢNG

```
        list = list->link;
    }
}
```

- Trong file **chuongtrinh\_lab5.cpp** như sau:

### Mã lệnh

```
#include "thuvien_lab5.h"
void main(){
    Nodeptr dssv;
    nhap_DSSV(dssv);
    xuat_DSSV(dssv);
    giaiPhong(dssv);
    system("pause");
}
```

**Kết quả xuất ra ở đoạn mã trên như sau:**

```
C:\WINDOWS\system32\cmd.exe

Nhap MaSV (NHAP 0 DE THOAT):sv01
Nhap Ten SV:Le Dung
Nhap Diem SV:8
Nhap MaSV (NHAP 0 DE THOAT):sv01
Ma so sv bi trung
Nhap lai ma so sv:sv02
Nhap Ten SV:Hoang Nam
Nhap Diem SV:7
Nhap MaSV (NHAP 0 DE THOAT):0
MASV      TEN SV      Diem
sv02      Hoang Nam    7
sv01      Le Dung     8
Press any key to continue . . .
```

**Câu 2:** Yêu cầu, sinh viên sử dụng đoạn mã trên để viết hàm thêm cuối sinh viên, gọi và chạy lại chương trình.

**Câu 3:** Cho hàm tìm kiếm một node trong DSLK:

**Mã lệnh**

```
Nodeptr tim_Node_X(Nodeptr list, int x){
    Nodeptr p = list;
    while (p != NULL){
        if (p->data == x)
            return p;
        p = p->link;
    }
    return p;
}
```

**Yêu cầu:**

- Chỉnh sửa hàm **tim\_node\_x** ở trên, để xuất ra thông tin của sinh viên vừa tìm được dựa vào mã sinh viên nhập vào, có nguyên mẫu hàm như sau: **Nodeptr tim\_Sv\_Masv(Nodeptr list, SinhVien x)**
- **Kết quả xuất ra như sau:**

```
C:\WINDOWS\system32\cmd.exe

Nhap MaSV (NHAP 0 DE THOAT):sv01
Nhap Ten SV:le dung
Nhap Diem SV:8
Nhap MaSV (NHAP 0 DE THOAT):sv02
Nhap Ten SV:Van Anh
Nhap Diem SV:9
Nhap MaSV (NHAP 0 DE THOAT):0
MASV      TEN SV      Diem
sv02      Van Anh      9
sv01      le dung      8

Nhap Masv can tim:sv01

Co sinh vien:sv01      le dung      8
Press any key to continue . . .
```

**Câu 4:** Đoạn mã sau đây là hàm xóa đầu (**xoa\_Dau**) của một danh sách.

Mã lệnh
<pre>void xoa_Dau(Nodeptr&amp; list){     Nodeptr p;     if (list != NULL){         p = list;         list = p-&gt;link;         delete p;     } }</pre>

**Yêu cầu:**

- a/. Sinh viên hãy gọi hàm xóa đầu (**xoa\_Dau(Nodeptr& list)**) vừa viết ở file cài đặt ở trên vào trong hàm **void main**. Xuất ra danh sách sinh viên còn lại sau khi gọi hàm xóa đầu.

## Bài 5: DANH SÁCH LIÊN KẾT ĐƠN – TRÊN ĐỐI TƯỢNG

b/. Sinh viên hãy viết thêm hàm xóa cuối (**xoa\_Cuoi(Nodeptr& list)**), sau đó gọi hàm xóa cuối vừa viết vào trong **void main** để thực thi chương trình.

**Câu 5:** Đoạn mã sau đây là hàm xóa một sinh viên khi biết mã sinh viên đó.

### Mã lệnh

```
void xoa_Sv_Masv(Nodeptr& list, SinhVien sv){
    if (strcmp(list->data.masv, sv.masv) == 0){
        Nodeptr p = list;
        list = list->link;
        delete p;
    }
    else{
        Nodeptr p, before;
        p = list;
        before = list;
        while (p != NULL && strcmp(p->data.masv, sv.masv)
!= 0){
            before = p;
            p = p->link;
        }
        if (p != NULL){
            before->link = p->link;
            delete p;
        }
    }
}
```

### Yêu cầu:

- a/. Viết lời gọi hàm **xoa\_Sv\_Masv** ở trên trong hàm main.
- b/. Viết hàm xóa tất cả các sinh viên có điểm <5 (**Bài tập nâng cao – về nhà**)

### Gợi ý:

- Sử dụng hàm tìm kiếm những sinh viên có điểm dưới

- Viết hàm tìm kiếm những sinh viên có điểm dưới 5
- Viết hàm xóa tất cả các sinh viên có điểm dưới 5

#### **IV. BÀI TẬP VỀ NHÀ**

**Câu 1:** Sinh viên hãy viết hàm đảo ngược một DSLK có nguyên mẫu hàm như sau: `void reverseList_SV(NODEPTR &list)`.  
viết thêm các lệnh cần thiết để chương trình có thể thực thi được.

**Ví dụ :**

**DS nhập vào là :**

<b>Mssv</b>	<b>Ten</b>	<b>Điểm</b>
A5130001	Nguyen Van Lai	8
A5130002	Tran Thi Mai	3
A5130003	Le Thi Dao	4

**DS đảo ngược là :**

<b>Mssv</b>	<b>Ten</b>	<b>Điểm</b>
A5130003	Le Thi Dao	4
A5130002	Tran Thi Mai	3
A5130001	Nguyen Van Lai	8

**Câu 2:** Sinh viên hãy viết hàm **SplitList\_SinhVien()** để tách một danh sách sinh viên thành hai danh sách sinh viên trong đó :

- Một danh sách sinh viên chứa các sinh viên có điểm nhỏ hơn 5
- Một danh sách sinh viên chứa các điểm lớn hơn hoặc bằng 5

Viết thêm các hàm cần thiết để minh họa hàm vừa viết thực thi được.

**Ví dụ :**

**DS nhập vào là :**

<b>Mssv</b>	<b>Tên</b>	<b>Điểm</b>
A5130001	Nguyen Van Lai	8
A5130002	Tran Thi Mai	3
A5130003	Le Thi Dao	4
A5130004	Ta Thi Cuc	5
A5130005	Ly Thi Hue	8
A5130006	Huynh Kiem Lan	9

**Tách thành 2 DS :****➤ DSSV có điểm số nhỏ hơn 5 :**

A5130002	Tran Thi Mai	3
A5130003	Le Thi Dao	4

**➤ DSSV có điểm số lớn hơn hoặc bằng 5 :**

A5130001	Nguyen Van Lai	8
A5130004	Ta Thi Cuc	5
A5130005	Ly Thi Hue	8
A5130006	Huynh Kien Lan	8

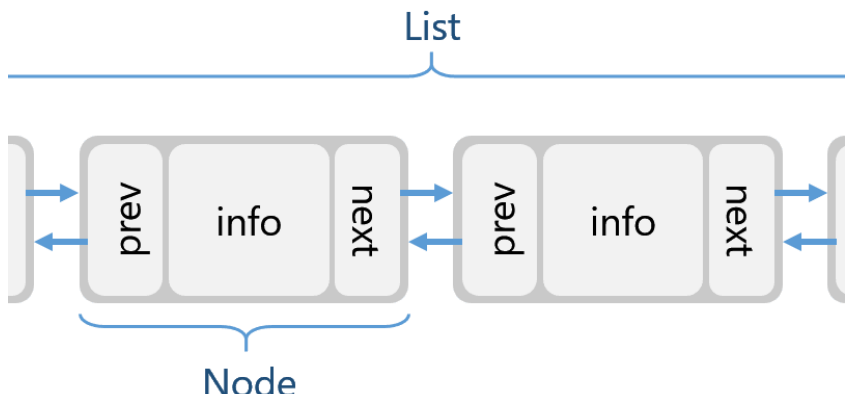
**Câu 3.** Nhập thêm vào cấu trúc sinh viên đã cho thông tin các điểm toán, lý, hóa. Yêu cầu tính điểm trung bình. Sắp xếp danh sách sinh viên theo chiều tăng dần của điểm trung bình.

## BÀI 6: DANH SÁCH LIÊN KẾT KÉP – DSLK VÒNG

### I. MỤC TIÊU:

- Sinh viên biết cách cài đặt danh sách liên kết kép (double Linked List)
- Sinh viên biết cách cài đặt danh sách liên kết vòng (circular Linked List)
- Xây dựng các phép toán cơ bản trên bằng danh sách liên kết kép và vòng

### II. TÓM TẮT LÝ THUYẾT

Nội dung	Ví dụ
<b>1. Danh sách liên kết KÉP (double linked list)</b> <ul style="list-style-type: none"> <li>• <b>Định nghĩa :</b> Danh sách liên kết kép cũng là một danh sách liên kết trong đó mỗi phần tử của nó liên kết với phần tử đứng trước và sau nó trong danh sách</li> <li>• <b>Cách biểu diễn:</b> Mỗi phần tử của danh sách liên kết đơn gồm 03 vùng: <ul style="list-style-type: none"> <li>- Vùng chứa dữ liệu (<i>info</i>)</li> <li>- Vùng chứa liên kết đến phần tử kế <i>next</i></li> <li>- Vùng chứa liên kết đến phần tử trước đó <i>pre</i></li> </ul> </li> <li>• <b>Hình ảnh minh họa:</b></li> </ul> 	
- Khai báo kiểu dữ liệu cho danh sách	<pre>#include&lt;iostream&gt; using namespace std;</pre>

số nguyên:	<pre> typedef int TYPEINFO; struct Node{     TYPEINFO data;     Node* next;     Node* pre; }; typedef Node* Nodeptr; struct Dlist{     Nodeptr first;     Nodeptr last; }; </pre>
- Khởi tạo danh sách liên kết	<pre> void khoi_Tao(Dlist&amp; list){     list.first = NULL;     list.last = NULL; } </pre>
- Hàm tạo một node:	<pre> Nodeptr tao_Node(TYPEINFO x){     Nodeptr p;     p = new Node;     p-&gt;data = x;     p-&gt;next = NULL;     p-&gt;pre = NULL;     return p; } </pre>
- Kiểm tra danh sách rỗng hay không	<pre> int isEmpty(Dlist list){     return (list.first == NULL ? 1 : 0); } </pre>
- Thêm 1 phần tử vào đầu danh sách liên kết kép:	<pre> void them_Dau(Dlist&amp; list, TYPEINFO x){     Nodeptr p;     p = tao_Node(x);     if (isEmpty(list)){         list.first = p;         list.last = p;     }     else{         p-&gt;next = list.first;         list.first-&gt;pre = p;         list.first = p;     } } </pre>



	<pre>         }     } </pre>
<p>- Thêm 1 phần tử vào giữa danh sách liên kết kép (Thêm sau phần tử q):</p>	<pre> void them_Sau_q(NODEPTR q, TYPEINFO x) {     NODEPTR p;     p = Create_Node(x);     p-&gt;next=q-&gt;next;     p-&gt;pre = q;     q-&gt;next=p;     q-&gt;next-&gt;pre=p; } </pre>
<p>- Thêm 1 phần tử vào cuối DSLK kép</p>	<pre> //them cuoi void them_Cuoi(Dlist&amp; l, TYPEINFO x) {     Nodeptr p;     p = tao_Node(x);     if (isEmpty(l)){         l.first = p;         l.last = p;     }     else{         l.last-&gt;next = p;         p-&gt;pre = l.last;         l.last = p;     } } </pre>
<p>- Đoạn mã xóa 1 phần tử ở đầu danh sách liên kết kép</p>	<pre> void Xoa_dau(Dlist &amp;l){     if (isEmpty(l) != NULL){         l.first = l.first-&gt;next;     } } </pre>

- Đoạn mã xóa phần tử cuối của danh sách liên kết kép

```
void xoa_Cuoi(Dlist& l){
    if (l.first != NULL)
        cout << "\n DS rong";
    else if (l.first == l.last){
        delete l.first;
        l.first = l.last = NULL;
    }
    else{
        Nodeptr p = l.last;
        l.last->pre->next = NULL;
        l.last = l.last->pre;
        delete p;
    }
}
```

## 2. DSLK VÒNG (circular linked list)

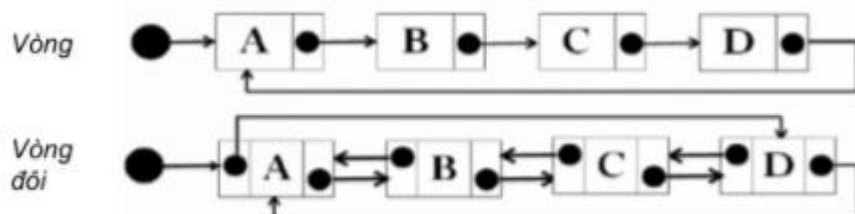
- **Định nghĩa** : Danh sách liên kết vòng (Circular Linked List) là một biến thể của danh sách liên kết (Linked List), trong đó phần tử đầu tiên trở tới phần tử cuối cùng và phần tử cuối cùng trở tới phần tử đầu tiên.

- Cả hai loại Danh sách liên kết đơn (Singly Linked List) và Danh sách liên kết đôi (kép - Doubly Linked List) đều có thể tạo thành dạng Danh sách liên kết vòng.

- **Cách biểu diễn**: Mỗi phần tử của danh sách liên kết đơn gồm 02 vùng:

- Vùng chứa dữ liệu data
- Vùng chứa liên kết đến phần tử kế next

- **Hình ảnh minh họa**:



- Chúng ta sẽ tìm hiểu danh sách liên kết vòng từ DSLK đơn.

- Khai báo kiểu dữ liệu cho danh sách số nguyên:

```
typedef int TYPEINFO;
struct Node{
    TYPEINFO data;
    Node* link;
};
typedef Node* Nodeptr;
struct List{
    Nodeptr first;
    Nodeptr last;
};
```

### III. NỘI DUNG THỰC HÀNH

**Câu 1:** Cho đoạn mã sau, thêm một phần tử vào DSLK kép, yêu cầu sinh viên nhập đoạn mã và biên dịch chương trình.

- Trong file **thuvien\_lab6.h** như sau:

#### Mã lệnh

```
#include<iostream>
using namespace std;
typedef int TYPEINFO;
struct Node{
    TYPEINFO data;
    Node* next;
    Node* pre;
};
typedef Node* Nodeptr;
struct Dlist{
    Nodeptr first;
    Nodeptr last;
};

void khoi_Tao(Dlist& list);
int isEmpty(Dlist L);
Nodeptr tao_Node(TYPEINFO x);
void them_Dau(Dlist& L, TYPEINFO x);
void nhap_Ds(Dlist& L);
```

```
void xuat_Ds_Thuan(Dlist list);  
void xuat_Ds_Nguoc(Dlist L);
```

- Trong file **caidat\_lab6.cpp** như sau:

**Mã lệnh**

```
#include "thuvien_lab6_kep.h"  
void khoi_Tao(Dlist& list){  
    list.first = NULL;  
    list.last = NULL;  
}  
int isEmpty(Dlist list){  
    return (list.first == NULL ? 1 : 0);  
}  
Nodeptr tao_Node(TYPEINFO x){  
    Nodeptr p;  
    p = new Node;  
    p->data = x;  
    p->next = NULL;  
    p->pre = NULL;  
    return p;  
}  
void them_Dau(Dlist& list, TYPEINFO x){  
    Nodeptr p;  
    p = tao_Node(x);  
    if (isEmpty(list)){  
        list.first = p;  
        list.last = p;  
    }  
    else{  
        p->next = list.first;  
        list.first->pre = p;  
        list.first = p;  
    }  
}  
void nhap_Ds(Dlist& list){  
    int x;  
    khoi_Tao(list);  
    do{
```

```
        cout << "\n Nhap x= (-99 de thoat)";
        cin >> x;
        if (x == -99)
            break;
        them_Dau(list, x);
    } while (1);
}
//Xuat theo chieu thuan
void xuat_Ds_Thuan(Dlist list){
    Nodeptr p = list.first;
    while (p != NULL){
        cout << p->data << " ";
        p = p->next;
    }
}
```

Trong file **chuongtrinh\_lab6.cpp** như sau:

#### Mã lệnh

```
#include"thuvien_lab6_kep.h"
void main(){
    Dlist list;
    khoi_Tao(list);
    nhap_Ds(list);
    cout << "\n Xuat danh sach theo chieu thuan:";
    xuat_Ds_Thuan(list);
}
```

#### **Yêu cầu:**

- a/. Nhập đoạn mã trên và chạy chương trình
- b/. Viết thêm hàm xuất (**Xuat\_ds\_nguoc**) nghĩa là xuất ra một danh sách chạy theo thứ tự ngược lại với hàm (**Xuat\_ds\_thuan**)

**Câu 2:** Đoạn mã sau là hàm thêm một phần tử vào danh sách liên kết kép ở vị trí cuối, yêu cầu sinh viên nhập thêm vào chương trình trên và chạy chương trình.

**Mã lệnh**

```
//them cuoi
void them_Cuoi(Dlist& l, TYPEINFO x)
{
    Nodeptr p;
    p = tao_Node(x);
    if (isEmpty(l)){
        l.first = p;
        l.last = p;
    }
    else{
        l.last->next = p;
        p->pre = l.last;
        l.last = p;
    }
}
```

**Câu 3:** Hãy viết hàm xóa đầu để xóa phần tử đầu danh sách liên kết kép, có nguyên mẫu hàm như sau: **void Xoa\_dau(DLIST &l)**

**Câu 4:** Hãy viết hàm xóa cuối để xóa phần tử cuối danh sách liên kết kép, có nguyên mẫu hàm như sau: **void Xoa\_cuoi(DLIST &l)**

**Câu 5:** Hãy viết hàm tìm kiếm một node có giá trị là x có trong danh sách liên kết kép, nếu có trả về vị trí tìm được, nếu không có xuất ra thông báo không có phần tử cần tìm, có nguyên mẫu hàm như sau: **int tim\_Vitri\_X(Dlist l, TYPEINFO x)**

**Câu 6:** Sau đây là đoạn mã về danh sách liên kết vòng, yêu cầu sinh viên nhập đoạn mã vào:

- Trong file **thuvien\_lab6\_dslkvong.h** như sau:

**Mã lệnh**

```
#include<iostream>
using namespace std;
typedef int TYPEINFO;
struct Node{
    TYPEINFO data;
    Node* link;
};
typedef Node* Nodeptr;
struct List{
    Nodeptr first;
    Nodeptr last;
};
Nodeptr tao_Node(TYPEINFO x);
void khoiTao(List &list);
int isEmpty(List list);
void them_Dau(List &list, TYPEINFO x);
void them_Cuoi(List &list, TYPEINFO x);
void readList(List &list);
void printList(List list);
void xoa_Dau(List &list);
```

- Trong file **caidat\_lab6\_dslkvong.cpp** như sau:

**Mã lệnh**

```
#include"thuvien_lab6_dslkvong.h"
Nodeptr tao_Node(TYPEINFO x){
    Nodeptr p = new Node;
    if (p == NULL)
        return 0;
    p->link = NULL;
    p->data = x;
    return p;
}

void khoiTao(List& list){
    list.first = list.last = NULL;
}
int isEmpty(List list){
```

```

        if (list.first == NULL)
            return 1;
        return 0;
    }

void them_Dau(List& list, TYPEINFO x){
    Nodeptr p = tao_Node(x);
    if (isEmpty(list) == 1)
        list.first = list.last = p;
    else{
        p->link = list.first;
        list.first = p;
    }
    list.last->link = list.first;
}

void readList(List& list){
    int x;
    do{
        cout << "\n Nhap x=(thoat-99)";
        cin >> x;
        if (x == -99)
            break;
        them_Dau(list, x);
        //Insert_last(list,x);
    } while (1);
}

void printList(List list){
    if (list.first != NULL){
        cout << "\n Cac phan tu cua DS Vong:";
        Nodeptr p = list.first;
        do {
            cout << p->data << " ";
            p = p->link;
        } while (p != list.first);
    }
    else
        cout << "\n DS rong";
}

```



- Trong file **chuongtrinh\_lab6\_dslkvong.cpp** như sau:

Mã lệnh
<pre>#include "thuvien_lab6_dslkvong.h" void main(){     List k;     khoiTao(k);     readList(k);     printList(k);     system("pause"); }</pre>

**Yêu cầu:**

- a/. Sinh viên hãy nhập đoạn mã trên và chạy chương trình.
- b/. Sinh viên hãy viết thêm hàm thêm cuối danh sách, sau đó gọi và chạy thử chương trình

**Câu 7:** Cho hàm xóa đầu của DSLK vòng như sau:

Mã lệnh
<pre>void xoa_Dau(List&amp; list){     if (list.first != NULL){         if (list.first != list.last){             Nodeptr p = list.first;             list.first = list.first-&gt;link;             list.last-&gt;link = list.first;             delete p;         }         else             list.first = NULL;     } }</pre>

**Yêu cầu:**

- a/. Sinh viên hãy nhập đoạn mã trên vào, gọi và chạy thử chương trình.
- b/. Sinh viên viết thêm hàm xóa cuối và chạy thử chương trình.

**Câu 8:** Sinh viên hãy viết hàm tìm kiếm node có giá trị bằng x có trong DSLK vòng.

## BÀI 7: HÀNG ĐỢI

### I. MỤC TIÊU:

Sau khi thực hành xong bài này, sinh viên phải biết cách:

- Khai báo queue dùng danh sách đặc
- Khai báo queue dùng danh sách liên kết
- Cài đặt các phép toán trên queue

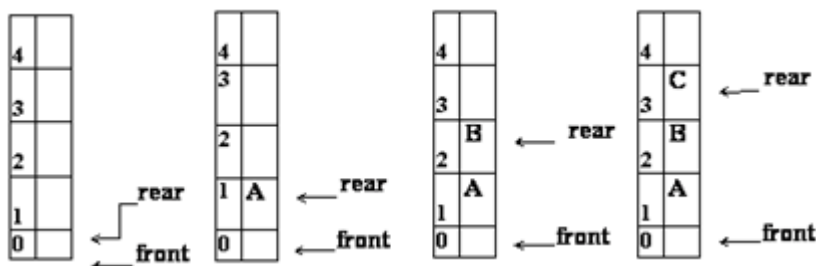
### II. TÓM TẮT LÝ THUYẾT

#### 1. Queue

- Một hàng (queue) là một cấu trúc dữ liệu mà việc thêm vào được thực hiện ở một đầu (rear) và việc lấy ra được thực hiện ở đầu còn lại (front).
- Queue làm việc theo cơ chế: Phần tử vào trước sẽ ra trước – FIFO (First In First Out)

- **Hình ảnh minh họa:**

- Biến đầu front chỉ ra vị trí thực hiện phép loại bỏ
- Biến cuối rear để chỉ ra vị trí thực hiện phép thêm vào.



Nội dung	queue – cài đặt DS Đặc	queue – Cài đặt DSLK
- Khai báo cấu trúc queue cho danh sách số nguyên:	<pre>struct Queue {     int front,     rear;     int     elem[MAX];</pre>	<pre>typedef struct NODE *NODEPTR; typedef int TYPEINFO; struct NODE {     int data;</pre>

	};	<pre>         NODE* link;     };     struct QUEUE     {         NODEPTR front,         rear;     }     ; </pre>
- Khởi tạo queue	<pre> void Init(Queue &amp;q) {     q.front = -1;     q.rear = -1; } </pre>	<pre> void Init(QUEUE &amp;q) {     q.front = NULL;     q.rear = NULL; } </pre>
- Kiểm tra danh sách rỗng hay không	<pre> int isEmpty(Queue q) {     if (q.front == -1)         return 1;     return 0; } </pre>	<pre> int IsEmpty(QUEUE q) {     if (q.front == NULL)         return 1;     return 0; } </pre>
- Kiểm tra Queue có đầy hay chưa	<pre> int isFull(Queue q) {     return (q.rear - q.front == MAX - 1); } </pre>	
- Tạo node	<pre> NODEPTR Tao_node(TYPEINFO x) {     NODEPTR p;     p = new NODE;     p-&gt;data = x;     p-&gt;link = NULL;     return p; } </pre>	

- Có 3 thủ tục chính trên Queue
  - **EnQueue:** Thêm một phần tử vào Queue
  - **DeQueue:** Lấy ra một phần tử khỏi Queue
  - **GetFront:** Xem phần tử đầu Queue

### III. NỘI DUNG THỰC HÀNH

#### Phần 1: Cài đặt Queue bằng danh sách Đặc

**Câu 1:** Sinh viên hãy nhập đoạn mã sau và chạy chương trình, kiểm tra chương trình có chạy được không? Nếu không, tìm lỗi và sửa lại cho đúng.

- Trong file **thuvien\_lab7.h** có đoạn mã như sau:

Mã lệnh
<pre> #include&lt;iostream&gt; //Khai bao so phan tu toi da cua queue const int MAX = 50; typedef int type; // khai bao cac hang so const int STOP = -99; using namespace std; struct Queue{     // front: phan tu dau tien     // rear: phan tu cuoi cung     int front, rear;     int elem[MAX]; }; void khoiTao(Queue&amp; q); int isEmpty(Queue q); int isFull(Queue q); int enqueue(Queue&amp; q, type x); void print(Queue q); void setValue(Queue&amp; q);         </pre>

- Trong file **caidat\_lab7.cpp** có đoạn mã như sau:

**Mã lệnh**

```
#include"thuvien_lab7.h"
void khoiTao(Queue& q){
    q.front = -1;
    q.rear = -1;
}
// Ham kiem tra Queue co rong hay khong
int isEmpty(Queue q){
    if (q.front == -1)
        return 1;
    return 0;
}
// Ham kiem tra Queue co day hay khong
int isFull(Queue q){
    return (q.rear - q.front == MAX - 1);
}
// Ham them mot phan tu vao hang doi. Hay sua(viet) lai ham nay cho chinh xac.
int enqueue(Queue& q, type x){
    int i;
    if (isFull(q))
        return 0;//Hang day
    else{
        if (q.front == -1)
            q.front = 0;
        else if (q.rear == MAX - 1){
            for (i = q.front; i <= q.rear; i++)
                q.elem[i - q.front] = q.elem[i];
            q.rear = MAX - 1 - q.front;
            q.front = 0;
        }
        q.rear++;
        q.elem[q.rear] = x;
        return q.rear;
    }
}
// In ham doi
void print(Queue q){
    if(isEmpty(q)!=-1){
```

```

        for (int i = q.front; i<=q.rear; i++){
            cout<<q.elem[i]<<" ";
        }
    }
    else
        cout<<"Rong!!!";
}
void setValue(Queue& q){
    int x;
    while (1){
        cout << "Nhap DL vao Queue(nhap -99 de thoat):
";
        cin >> x;
        if (x == STOP) break;
        enqueue(q, x);
    }
}

```

- Trong hàm **enqueue** dùng để thêm một phần tử vào queue. Hàm này có lỗi sai, sinh viên hãy sửa lại cho đúng.

- Trong file **chuongtrinh\_lab7.cpp** có đoạn mã như sau:

#### Mã lệnh

```

#include"thuvien_lab7.h"
void main(){
    Queue q;
    khoiTao(q);
    setValue(q);
    print(q);
}

```

**Câu 2:** Sau đây là đoạn mã lấy một phần tử ra khỏi queue, sinh viên nhập đoạn mã vào và gọi hàm trong void main để chạy chương trình.

#### Mã lệnh

```

// ham lay mot phan tu ra khoi ham doi theo quy tac (First
In First Out)
// ham tra ve phan tu duoc lay
int dequeue(Queue& q){

```

```

        if (isEmpty(q) != -1){
            if (q.front == q.rear)
                khaiTao(q);
            else{
                int t = q.elem[q.front];
                q.front = q.front + 1;
                return t;
            }
        }
        else
            return 0;
    }
}

```

**Câu 3:** Sinh viên tự viết đoạn mã dùng để xem giá trị của phần tử ở đầu hàng đợi, sau đó chạy chương trình.

## **Phần 2: Cài đặt Queue bằng danh sách liên kết**

**Câu 1:** Cho đoạn mã sau:

- Trong file **thuvien\_lab7\_dslk.h** có đoạn mã:

### **Mã lệnh**

```

#include<iostream>
using namespace std;
typedef struct Node* Nodeptr;
typedef int TYPEINFO;
struct Node{
    int data;
    Node* link;
};
struct Queue{
    Nodeptr front, rear;
};
void khaiTao(Queue& q);
int isEmpty(Queue q);
Nodeptr tao_Node(TYPEINFO x);
void enqueue(Queue& q, TYPEINFO x);
void setValue(Queue& q);
void print(Queue q);

```

- Trong file **caidat\_lab7\_dslk.cpp** có đoạn mã như sau:



Mã lệnh

```
#include "thuvien_lab7_dslk.h"
const int STOP = -99;
void khoiTao(Queue &q){
    q.front = NULL;
    q.rear = NULL;
}
int isEmpty(Queue q){
    if (q.front == NULL)
        return 1;
    return 0;
}
Nodeptr tao_node(TYPEINFO x){
    Nodeptr p;
    p = new Node;
    p->data = x;
    p->link = NULL;
    return p;
}
void EnQueue(Queue &q, TYPEINFO x){
    Nodeptr p;
    p = tao_node(x);
    if (isEmpty(q)){
        q.front = p;
        q.rear = p;
    }
    else{
        q.rear->link = p;
        q.rear = p;
    }
}
void setValue(Queue &q){
    int x;
    while (1){
        cout << "\n Nhập gia tri cho Queue:(Nhập -99) để
dung";
        cin >> x;
        if (x == STOP) break;
        EnQueue(q, x);
    }
}
```

```
void print(Queue q){
    Nodeptr p;
    p = q.front;
    while (p != NULL){
        cout << p->data << " ";
        p = p->link;
    }
}
```

- Trong file **chuongtrinh\_lab7\_dslk.cpp** có đoạn mã như sau:

#### Mã lệnh

```
#include"thuvien_lab7_dslk.h"
void main(){
    Queue q;
    khoiTao(q);
    setValue(q);
    print(q);
    system("pause");
}
```

- Các bạn nhập đoạn mã trên và chạy chương trình cài đặt queue bằng DSLK.

**Câu 2:** Sau đây là hàm DeQueue, lấy một phần tử ra khỏi hàng đợi, sinh viên hãy nhập vào và gọi hàm trong **void main**.

#### Mã lệnh

```
TYPEINFO deQueue(Queue &q){
    if (isEmpty(q))
        return -1;
    else{
        Nodeptr p = q.front;
        q.front = p->link;
        TYPEINFO t = p->data;
        delete p;
        return t;
    }
}
```

**Câu 3:** Sinh viên tự viết hàm **TYPEINFO** **getFront(QUEUE q)** và gọi hàm trong void main, để xem phần tử đầu hàng đợi.

## BÀI 8: NGĂN XẾP

### I. MỤC TIÊU:

Sau khi thực hành xong bài này, sinh viên phải biết cách:

- Khai báo một ngăn xếp
- Cài đặt các phép toán trên ngăn xếp dùng danh sách đặc
- Cài đặt các phép toán trên ngăn xếp dùng danh sách liên kết

### II. TÓM TẮT LÝ THUYẾT

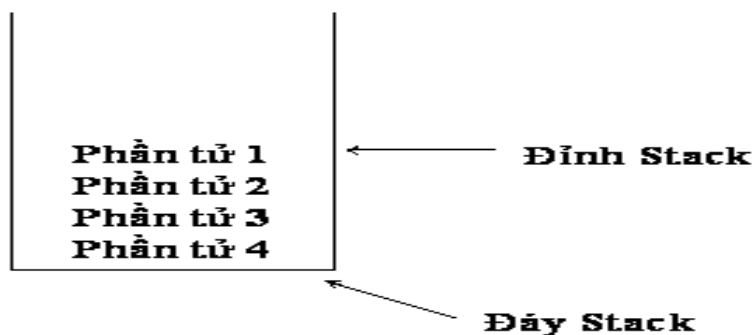
#### 1. Stack (ngăn xếp)

- **Định nghĩa** : Một ngăn xếp (stack) là một cấu trúc dữ liệu mà việc thêm vào và loại bỏ

được thực hiện tại một đầu (gọi là đỉnh – top của stack).

- Một ngăn xếp đĩa đặt trên bàn cho ta hình ảnh trực quan của ngăn xếp, muốn thêm vào ngăn xếp đó 1 đĩa ta để đĩa mới trên đỉnh ngăn xếp, muốn lấy các đĩa ra khỏi ngăn xếp ta cũng phải lấy đĩa trên trước.
- Như vậy ngăn xếp là một cấu trúc có tính chất "vào sau - ra trước" hay LIFO (last in - first out ).

- **Cách biểu diễn:**



Nội dung	Stack – cài đặt danh sách đặc	Stack – Cài đặt danh sách liên kết
----------	-------------------------------	------------------------------------

- Khai báo cấu trúc Stack cho danh sách số nguyên:	<pre>#define MAXSTACK 100 typedef int TYPE; struct Stack{     int sp;     TYPE elem[MAXSTACK]; };</pre>	<pre>typedef int TYPEINFO; struct Node{     TYPEINFO data;     Node* link; }; typedef Node* Nodeptr; struct Stack{     Nodeptr top; } ;</pre>
- Khởi tạo stack	<pre>void Init_stack(Stack &amp;s){     s.sp=-1; }</pre>	<pre>void init_Stack (Stack&amp; s){     s.top=NULL; }</pre>
- Kiểm tra danh sách rỗng hay không	<pre>int isEmpty(Stack s){     return s.sp==-1; }</pre>	<pre>int isEmpty(Stack s){     return s.top==NULL; }</pre>
- Kiểm tra Stack có đầy hay chưa	<pre>int isFull(Stack s){     return s.sp==MAXSTACK-1; }</pre>	
- Tạo node		<pre>NODEPTR tao_Node(TYPEINFO x){     Nodeptr p;     p = new Node;     p-&gt;data=x;     p-&gt;link=NULL;     return p; }</pre>

- Có 3 thủ tục chính trên Stack
  - **push:** Thêm một phần tử vào ngăn xếp
  - **pop:** Lấy ra một phần tử khỏi ngăn xếp
  - **top:** Xem phần tử đầu ngăn xếp

### III. NỘI DUNG THỰC HÀNH

#### Phần 1: Cài đặt Stack bằng danh sách đặc

##### Câu 1: Sinh viên nhập đoạn mã sau vào:

- Trong file **thuvien\_lab8.h** có đoạn mã như sau, hãy nhập vào bài của bạn:

#### Mã lệnh

```
#include<iostream>
using namespace std;
#define MAXSTACK 100
typedef int TYPE;
struct Stack{
    int sp;
    TYPE elem[MAXSTACK];
};
void init_Stack(Stack& s);
int isEmpty(Stack s);
int isFull(Stack s);
void push(Stack& s, int x);
int pop(Stack& s);
void input_Stack(Stack& s);
void output_Stack(Stack s);
void get_Stack(Stack& s);
```

- Trong file **caidat\_lab8.cpp** có đoạn mã như sau, hãy nhập vào bài của bạn:

**Mã lệnh**

```
#include "thuvien_lab8.h"
const int stop = -99;
void init_Stack(Stack& s){
    s.sp=-1;
}
int isEmpty(Stack s){
    return s.sp==-1;
}
int isFull(Stack s){
    return s.sp==MAXSTACK-1;
}
void push(Stack& s, int x){
    if(isFull(s))
        cout<<"\n Ngan xep day";
    else{
        s.sp++;
        s.elem[s.sp]=x;
    }
}
int pop(Stack& s){
    int a;
    if(isEmpty(s))
        return -1;
    else{
        a = s.elem[s.sp];
        s.elem[s.sp--];
    }
    return a;
}
void output_Stack(Stack s){
    for(int i=0;i<=s.sp;i++)
        cout<<s.elem[i]<<" ";
}
```

- Trong file **chuongtrinh\_lab8.cpp** có đoạn mã như sau, hãy nhập vào bài của bạn:

**Mã lệnh**

```
#include "thuvien_lab8.h"
void main(){
    Stack s;
    init_Stack(s);
    input_Stack(s);
    output_Stack(s);
}
```

**Yêu cầu:**

- Viết đoạn mã trên vào chương trình của bạn, viết thêm hàm nhập vào một dãy số vào ngăn xếp, có nguyên mẫu hàm:

**void input\_Stack(Stack& s); .**

**Câu 2:** Cho đoạn mã lấy phần tử ra khỏi stack

**Mã lệnh**

```
int pop(Stack& s){
    int a;
    if(isEmpty(s))
        return -1;
    else{
        a = s.elem[s.sp];
        s.elem[s.sp--];
    }
    return a;
}
```

**Yêu cầu:**

- a/. Hãy nhập đoạn mã trên vào chương trình
- b/. Gọi hàm vừa viết trong void main, xuất thông tin phần tử vừa lấy ra và danh sách các phần tử còn lại có trong stack.

**Câu 3:** Sinh viên tự viết hàm **int top(STACK s)** để xuất ra thông tin của phần tử nằm ở đầu ngăn xếp.

**Phần 2: Cài đặt ngăn xếp bằng danh sách liên kết**

**Câu 1:** Sinh viên nhập đoạn mã sau và chạy chương trình



- Trong file **thuvien\_lab8\_dslk.h** có đoạn mã như sau, hãy nhập vào bài của bạn:

Mã lệnh
<pre>#include&lt;iostream&gt; using namespace std; typedef int TYPEINFO; struct Node{     TYPEINFO data;     Node* link; }; typedef Node* Nodeptr; struct Stack{     Nodeptr top; }; void init_Stack(Stack&amp; s); int isEmpty(Stack s); Nodeptr tao_Node(TYPEINFO x); void push(Stack &amp;s, TYPEINFO a); int pop(Stack &amp;s); int top(Stack &amp;s); void input_Stack(Stack &amp;s); void output_Stack(Stack s);</pre>

- Trong file **caidat\_lab8\_dslk.cpp** có đoạn mã như sau, hãy nhập vào bài của bạn:

Mã lệnh
<pre>#include"thuvien_lab8_dslk.h" const int stop = -99; void init_Stack (Stack &amp;s){     s.top=NULL; } int isEmpty(Stack s){     return s.top==NULL; } Nodeptr tao_Node(TYPEINFO x){     Nodeptr p;</pre>

```

        p = new Node;
        p->data=x;
        p->link=NULL;
        return p;
    }
    void push(Stack &s, TYPEINFO a){
        Nodeptr p;
        p=tao_Node(a);
        if(isEmpty(s))
            s.top =p;
        else{
            p->link=s.top;
            s.top=p;
        }
    }
    void input_Stack(Stack& s){
        int x;
        while(1){
            cout<<"\n Nhap gia tri cho Stack (Nhap -99) de
dung";
            cin>>x;
            if(x==stop)
                break;
            push(s,x);
        }
    }
    void output_Stack(Stack s){
        Nodeptr p=s.top;
        while(p!=NULL){
            cout<<p->data<<" ";
            p=p->link;
        }
    }

```

- Trong file **chuongtrinh\_lab8\_dslk.cpp** có đoạn mã như sau, hãy nhập vào bài của bạn:

**Mã lệnh**

```

#include"thuvien_lab8_dslk.h"
void main(){
    Stack s;
    init_Stack(s);

```

```

    input_stack(s);
    output_stack(s);
}

```

**Yêu cầu:**

- Nhập đoạn mã trên và chạy chương trình

**Câu 2:** Cho đoạn mã, lấy ra một phần tử khỏi ngăn xếp như sau:

**Mã lệnh**

```

int pop(Stack& s)
{
    Nodeptr p;
    TYPEINFO a;
    if (!isEmpty(s))
    {
        p = s.top;
        s.top = p->link;
        a = p->data;
        delete p;
        return a;
    }
}

```

**Yêu cầu:**

- Nhập đoạn mã trên vào chương trình
- Gọi trong **void main**, xuất ra thông tin của phần tử vừa lấy ra khỏi ngăn xếp, và xuất ra danh sách các phần tử còn lại có trong ngăn xếp.

**Câu 3:** Sinh viên tự viết hàm **int Top(STACK &s)** để xem thông tin phần tử đầu stack.

## BÀI 9: CÂY NHỊ PHÂN – CÂY NHỊ PHÂN TÌM KIẾM

### I. MỤC TIÊU:

- Sau khi học xong bài này sinh viên tìm hiểu về cây nhị phân.
- Hiểu và cài đặt được cây nhị phân tìm kiếm

### II. TÓM TẮT LÝ THUYẾT:

#### 1. Cây nhị phân (*binary tree*)

- Là cây có bậc 2

##### Bậc:

- *Bậc của cây* : số cây con tối đa của một nút trên cây. Cây nhị phân có bậc là 2.
- *Bậc của nút* : số nút con của nút đó

##### Mức:

- Node gốc ở mức 0.
- Node gốc của các cây con của một node ở mức  $m$  là  $m+1$ .

##### Chiều cao:

- Cây rỗng có chiều cao là 0.
- Chiều cao của cây là chiều cao lớn nhất của một trong hai cây con cộng thêm một
- (Hoặc: mức lớn nhất của các node cộng 1)

##### Đường đi (*path*)

- Tên các node có được trong quá trình đi từ node gốc theo các cây con đến một node nào đó.

##### Node trước, sau, cha, con:

- Node  $x$  là trước node  $y$  (node  $y$  là sau node  $x$ ), nếu trên đường đi đến  $y$  có  $x$ .

- Node x là cha node y (node y là con node x), nếu trên đường đi đến y node x nằm ngay trước node y.

**Node lá, trung gian:**

- Node lá là node không có cây con nào. Bậc của nút lá là 0.
- Node trung gian không là node gốc hay node lá. Bậc của nút trung gian là khác 0.

**Phép duyệt cây**

- Duyệt qua từng node của cây (mỗi node chỉ được duyệt qua một lần)

**Cách duyệt:**

- Gọi: N(node), L(left), R(right)
- Chính thức: NLR, LNR, LRN, NRL, RNL, RLN
- Chuẩn: NLR (preorder), LNR (inorder), LRN (postorder)

**2. Cây nhị phân tìm kiếm (Binary Search Tree – BST)**

- Một cây nhị phân tìm kiếm (BST) là một cây nhị phân rỗng hoặc mỗi node của cây này có các đặc tính sau:
  - Khóa của node gốc lớn hơn khóa của tất cả các node của cây con bên trái và nhỏ hơn khóa của tất cả các node của cây con bên phải.
  - Các cây con (bên trái, phải) là BST
- **Các tính chất khác của BST**
  - Node cực trái (hay cực phải):
    - Xuất phát từ node gốc
    - Đi sang trái (hay phải) đến khi không đi được nữa
  - Khóa của node cực trái là nhỏ nhất trong BST
  - Khóa của node cực phải là lớn nhất trong BST
  - BST là cây nhị phân có tính chất:
    - Khóa của node gốc lớn hơn khóa của node cực trái

- Khóa của node gốc nhỏ hơn khóa của node cực phải

Nội dung	Nội dung
- Khai báo cấu trúc Cây nhị phân tìm kiếm cho danh sách số nguyên:	<pre>typedef int TYPEINFO; struct Node{     TYPEINFO data;     Node* left;     Node* right; }; typedef Node* Nodeptr; typedef Nodeptr BST;</pre>
- Khởi tạo Cây	<pre>void init_Tree(BST &amp;root){     root = NULL; }</pre>
- Tạo node	<pre>Nodeptr tao_Node(int x){     Nodeptr p;     p = new Node;     p-&gt;data = x;     p-&gt;left = NULL;     p-&gt;right = NULL;     return p; }</pre>
- Thêm phần tử vào cây nhị phân tìm kiếm	<pre>void insertTree(BST &amp;root, int x){     if (root == NULL)         root = tao_Node(x);     else{         if (x == root-&gt;data){             cout &lt;&lt; "\n Co node roi";             return;         }         else             if (x &lt; root-&gt;data)                 insertTree(root-&gt;left, x);             else                 insertTree(root-&gt;right, x);     } }</pre>

	}
- Tìm node a có tồn tại trong cây BST	<pre> Nodeptr tim_Node(BST tree, TYPEINFO a){     Nodeptr p = tree;     while (p != NULL){         if (p-&gt;data == a)             return p;         else{             if (p-&gt;data&gt;a) p = p-&gt;left;             else                 p = p-&gt;right;         }     }     return p; } </pre>

### III. NỘI DUNG THỰC HÀNH

**Câu 1:** Sinh viên nhập đoạn mã sau và chạy chương trình

- Trong file **thuvien\_lab8.h** có đoạn mã như sau, hãy nhập vào bài của bạn:

Mã lệnh
<pre> #include&lt;iostream&gt; #include&lt;windows.h&gt; using namespace std; typedef int TYPEINFO; struct Node{     TYPEINFO data;     Node* left;     Node* right; }; typedef Node* Nodeptr; typedef Nodeptr BST; Nodeptr tao_Node(int x); void init_Tree(BST &amp;root); void gotoxy(short x, short y); </pre>

```
void insertTree(BST &root, int x);
void nhapCay(BST &root);
void NLR(BST T, int x, int y, int d);
int tongSoNut(BST t);
int tongNutLa(BST t);
int chieuCao(BST tree);
void traverse_NLR(BST &root);
void traverse_LNR(BST &root);
void traverse_LRN(BST &root);
int tongSoNut(BST t);
int tongNutLa(BST t);
int chieuCao(BST tree);
int muc_Cay(BST root, TYPEINFO x);
Nodeptr tim_Node(BST tree, TYPEINFO a);
```

- Trong file **caidat\_lab9.cpp** có đoạn mã như sau, hãy nhập vào bài của bạn:

**Mã lệnh**

```
#include "thuvien_lab9.h"
Nodeptr tao_Node(int x){
    Nodeptr p;
    p = new Node;
    p->data = x;
    p->left = NULL;
    p->right = NULL;
    return p;
}
void init_Tree(BST &root){
    root = NULL;
}
void gotoxy(short x, short y){
    HANDLE hCon = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD pos;
    pos.X = x - 1;
    pos.Y = y - 1;
    SetConsoleCursorPosition(hCon, pos);
}
void insertTree(BST &root, int x){
```



```

    if (root == NULL)
        root = tao_Node(x);
    else{
        if (x == root->data){
            cout << "\n Co node roi";
            return;
        }
        else
            if (x<root->data)
                insertTree(root->left, x);
            else
                insertTree(root->right, x);
        }
    }
}
void nhapCay(BST &root){
    int x;
    while (1){
        cout << "\n Nhap DL cho cay (thoat nhan -99)";
        cin >> x;
        if (x == -99)
            break;
        insertTree(root, x);
    }
}
void NLR(BST T, int x, int y, int d){
    if (T != NULL){
        gotoxy(x, y);
        cout << "(" << T->data << ")";

        if (T->left != NULL){
            gotoxy(x + 1, y + 1);
            cout << '|';
            for (int i = x - d / 2 + 2; i <= x + 2;
i++) {
                gotoxy(i, y + 2);
                cout << '-';
            }
            NLR(T->left, x - d / 2, y + 3, d / 2);
        }
        if (T->right != NULL){
            gotoxy(x + 1, y + 1);

```

```

        cout << '|';
        for (int i = x + 2; i <= x + d / 2 + 2;
i++){
            gotoxy(i, y + 2);
            cout << '-';
        }
        NLR(T->right, x + d / 2, y + 3, d / 2);
    }
}
}

```

- Trong file **chuongtrinh\_lab9.cpp** có đoạn mã như sau, hãy nhập vào bài của bạn:

**Mã lệnh**

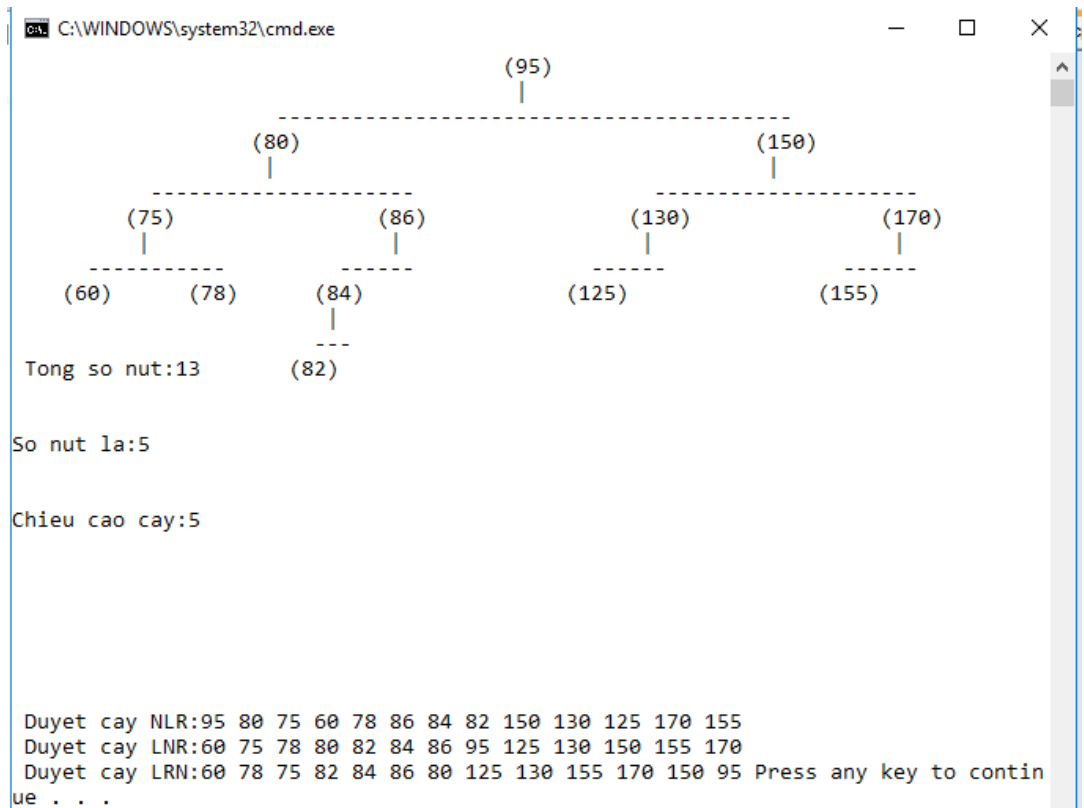
```

#include"thuvien_lab9.h"
void main(){
    BST tree;
    init_Tree(tree);
    nhapCay(tree);
    system("cls");

    cout << " \n\n\n\n\n\n\n\n\n\n\n\n\n\n";
    cout << " \n\n\n\n\n\n\n\n\n\n\n\n\n\nCÂY BAN VUA TAO
LA:";
    cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n";
    NLR(tree, 40, 1, 40);
}

```

- Khi ta nhập vào dãy số sau: 95, 80, 150, 170, 130, 86, 75, 78, 84, 95, 82, 60, 125, 155. Kết quả xuất cây nhị phân tìm kiếm như sau:



**Câu 2:** Các bạn viết thêm các hàm tính tổng số nút có trên cây, số nút lá, và xuất ra chiều cao cây như kết quả trên.

**Câu 3:** Viết 3 hàm duyệt cây BST: NLR, LNR, LRN

**Câu 4:** Viết hàm kiểm tra xem node có giá trị là x có tồn tại trên cây nhị phân hay không, nếu có trả về vị trí của node đó, nếu không trả về NULL.

**Câu 5:** Viết hàm xóa một node trên cây nhị phân tìm kiếm.