

Mục lục

LỜI NÓI ĐẦU	1
LAB 01. Cài đặt một số lược đồ thuật giải	3
LAB 02. Thuật giải tìm kiếm trong	15
LAB 03. Thuật giải sắp xếp trong.....	34
LAB 04. Danh sách liên kết đơn : Các thao tác cơ bản	49
LAB 05. Danh sách liên kết đơn : Các phép toán tập hợp	95
LAB 05. Danh sách liên kết đơn : Các phép toán trên đa thức rời rạc.....	99
LAB 06. Danh sách liên kết đơn : Ngăn xếp (Stack)	107
LAB 06. Danh sách liên kết đơn : Hàng đợi (Queue)	116
LAB 07. Danh sách liên kết đơn vòng.....	124
LAB 08. Cây nhị phân - cây nhị phân tìm kiếm	138
LAB 09. Cân bằng (AVL)	169
ÔN TẬP CUỐI KỲ	176

LỜI NÓI ĐẦU

Hệ thống bài lab đi kèm theo giáo trình Cấu trúc dữ liệu và thuật giải 1 giúp sinh viên :

- Cài đặt cấu trúc dữ liệu trong giáo trình
- Cài đặt các thuật giải trên cấu trúc dữ liệu tương ứng
- Ứng dụng các cấu trúc dữ liệu và thuật giải trong giáo trình cho các dự án phù hợp.

Hệ thống bài lab bao gồm 9 bài :

- Lab 01. Cài đặt một số lược đồ thuật giải.
- Lab 02. Cài đặt các thuật giải tìm kiếm trong và ứng dụng.
- Lab 03. Cài đặt các thuật giải sắp xếp trong và ứng dụng :
 - Các thuật giải trực tiếp, đơn giản,
 - Các thuật giải nâng cao.
- Lab 04. Cài đặt danh sách liên kết đơn : Cấu trúc dữ liệu và các thao tác cơ bản,
- Lab 05. Cài đặt các phép toán tập hợp và đa thức rời rạc,
- Lab 06. Cài đặt ngăn xếp (Stack), hàng đợi (Queue) và các ứng dụng.
- Lab 07. Cài đặt danh sách liên kết vòng và ứng dụng.
- Lab 08. Cài đặt cây nhị phân tìm kiếm (BST) và ứng dụng.
- Lab 09. Cài đặt cây AVL.
- Lab 00. (phụ lục)
 - Ôn tập lập trình cấu trúc với các cấu trúc dữ liệu cơ bản.
 - Ôn tập nhập xuất các tập tin văn bản .

Kế hoạch thực tập (mỗi buổi 3 tiết) :

Buổi 1	Buổi 2	Buổi 3	Buổi 4	Buổi 5	Buổi 6	Buổi 7	Buổi 8	Buổi 9	Buổi 10
Lab 1	Lab 2	Lab 3	Lab 3	Lab 4	Lab 5	Lab 6	Lab 7	Lab 8	- Lab 9 - Ôn tập

Các tác giả

Nguyễn Văn Phúc

Nguyễn thị Thanh Bình

Trần Tuấn Minh

Đinh Viết Tuấn

Nguyễn Thị Lương

LAB 01. Cài đặt một số lược đồ thuật giải

A. Mục tiêu

Tìm hiểu và cài đặt các thuật giải chia để trị, quay lui.

B. Yêu cầu :

- Sinh viên tự ôn tập (mục C)

- Buổi thực hành 1 (3 t) :

- Sinh viên thực tập trong phòng lab :
 - **Thời gian : 2 tiết đầu**
 - Khối lượng và nội dung : Các bài trong phần D (luyện tập)
 - Yêu cầu lưu trữ
 - ✓ Tạo thư mục **MaSV_HoVaTen_Lab01** chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
- Sinh viên tiếp tục thực tập trong phòng lab :
 - **Thời gian : 1 tiết (cuối)**
 - Khối lượng và nội dung : Bài 1 E (bài tập)
 - Yêu cầu lưu trữ
 - ✓ Tạo project riêng tương ứng
 - ✓ xóa thư mục debug trong project
 - ✓ Lưu trữ trong thư mục trên (**MaSV_HoVaTen_Lab01**)
 - ✓ Nén thư mục
 - Hình thức thu bài
 - ✓ Giáo viên sẽ thu bài (tập tin nén trên) qua hệ thống thu bài trực tuyến tại phòng lab sau khi hết thời gian qui định.

C. Ôn tập

1. Thuật giải đệ quy :

- Lược đồ :

```
Recursion() ≡  
{  
    if(điều kiện dừng)  
        Xử lý trường hợp đặc biệt;  
    else  
        if(điều kiện hợp lệ)  
            Xử lý đệ qui;  
            //Liên hệ đệ qui  
}
```

2. Thuật giải chia để trị :

- Lược đồ :

Nếu gọi $D\&C(\mathfrak{R})$ - Với \mathfrak{R} là miền dữ liệu - là hàm thể hiện cách giải bài toán theo phương pháp chia để trị thì ta có thể viết :

```
void D\&C(\mathfrak{R})  
{  
    If (\mathfrak{R} đủ nhỏ)  
        giải bài toán;  
    Else  
    {  
        Chia \mathfrak{R} thành \mathfrak{R}1, ..., \mathfrak{R}m;  
        for (i = 1; i <=m; i++)  
            D\&C(\mathfrak{R}i);  
        Tổng hợp kết quả;  
    }
```

```
}  
}
```

3. Thuật giải quay lui

- Lược đồ :

Với n là số bước cần phải thực hiện, k là số khả năng mà x_i có thể chọn lựa, $Try(i)$ là bước thứ i để xác định x_i

```
Try(i) =  
for ( j = 1 → k)  
  If (  $x_i$  chấp nhận được khả năng j)  
  {  
    Xác định  $x_i$  theo khả năng j;  
    Ghi nhận trạng thái mới;  
    if(  $i < n$ )  
      Try(i+1);  
    else  
      Ghi nhận nghiệm;  
      Trả lại trạng thái cũ cho bài toán;  
  }
```

D. LUYỆN TẬP

Bài 1. (Bài toán tìm min, max)

Tìm giá trị min, max trong đoạn $a[l..r]$ của mảng $a[0..n-1]$.

Bước 1 : Tạo project đặt tên : Bai1_D_cdt gồm các tập tin rỗng : thuvien.h và program.cpp.

Bước 2 :

Trong tập tin program.cpp, soạn phần code tối thiểu để chạy được chương trình nhằm kiểm tra các phần sẽ thêm vào chương trình.

```
#include <iostream>  
#include <stdlib.h>  
#include <time.h>  
using namespace std;  
#include "Thuvien.h"  
void ChayChuongTrinh();  
int main()  
{  
  ChayChuongTrinh();  
  return 1;  
}  
void ChayChuongTrinh()  
{  
  system("PAUSE");  
}
```

Bước 3 : Trong tập tin thuvien.h, ta soạn phần code thực hiện giải quyết bài toán.

Để dễ xử lý lỗi, khi viết thêm một hàm nào thì ta chạy ngay chương trình để kiểm tra lỗi.

- Nội dung tập tin thuvien.h :

```
//Khai bao hang  
void NhapDay(int a[MAX], int n)  
{  
  int i;  
  srand((unsigned)time(NULL));  
  for (i = 0; i < n; i++)  
  {  
    a[i] = -8 + rand() % 17; // trong khoảng [-8,8]  
  }  
}
```

```
void XuatDay(int a[MAX], int n)
{
    for (int i = 0; i < n; i++)
        cout << a[i] << '\t';
}
void MinMax(int a[MAX], int l, int r, int &min, int &max)
{
    int min1, min2, max1, max2;
    if (l == r)
    {
        min = a[l];
        max = a[l];
    }
    else
    {
        MinMax(a, l, (l + r) / 2, min1, max1);
        MinMax(a, (l + r) / 2 + 1, r, min2, max2);
        if (min1 < min2)
            min = min1;
        else
            min = min2;
        if (max1 > max2)
            max = max1;
        else
            max = max2;
    }
}
```

Bước 4:

Trong Program.cpp, cập nhật lại hàm ChayChuongTrinh() :

```
void ChayChuongTrinh()
{
    int a[MAX], n;
    int l, r;
    int min = 0, max = 0;
    cout << "\nn = "; cin >> n;
    NhapDay(a, n);
    cout << "\nl = "; cin >> l;
    cout << "\nr = "; cin >> r;
    MinMax(a, l, r, min, max);
    cout << "\nDay da cho:\n";
    XuatDay(a, n);
    cout << "\nMina[" << l << ",..." << r << "] = " << min;
    cout << "\nMaxa[" << l << ",..." << r << "] = " << max;
    system("PAUSE");
}
```

Chạy chương trình kiểm tra kết quả.

Kết thúc chương trình.

Bài 2. (Bài toán hoán đổi 2 phần trong 1 dãy.)

$a[1..n]$ là một mảng gồm n phần tử. Ta cần chuyển m ($1 \leq m \leq n$) phần tử đầu tiên của mảng với phần còn lại của mảng ($n-m$ phần tử) mà không dùng một mảng phụ.

Chẳng hạn, với $n = 8, a[8] = (1, 2, 3, 4, 5, 6, 7, 8)$

Nếu $m = 3$, thì kết quả là : $(4, 5, 6, 7, 8, 1, 2, 3)$

Nếu $m = 5$, thì kết quả là : $(6, 7, 8, 1, 2, 3, 4, 5)$

Nếu $m = 4$, thì kết quả là : $(5, 6, 7, 8, 1, 2, 3, 4)$

Bước 1 : Tạo project đặt tên : Bai2_D_cdt gồm các tập tin rỗng : thuvien.h và program.cpp.

Bước 2 :

Trong tập tin program.cpp, soạn phần code tối thiểu để chạy được chương trình nhằm kiểm tra các phần sẽ thêm vào chương trình.

```
#include <iostream>
using namespace std;
#include "Thuvien.h"
void ChayChuongTrinh();
int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    system("PAUSE");
}
Chạy chương trình kiểm tra lỗi.
```

Bước 3 : Trong tập tin thuvien.h, ta soạn phần code thực hiện giải quyết bài toán.

Để dễ xử lý lỗi, khi viết thêm một hàm nào thì ta chạy ngay chương trình để kiểm tra lỗi.

- Nội dung tập tin thuvien.h :

```
#define MAX 100

void NhapDay(int a[MAX], int n)
{
    int i;
    for (i = 1; i <= n; i++)
    {
        a[i] = i;
    }
}

void XuatDay(int a[MAX], int n)
{
    for (int i = 1; i <= n; i++)
        cout << a[i] << '\t';
}

void HoanVi(int &x, int &y)
{
    int t = x;
    x = y;
    y = t;
}

void Exchange(int a[MAX], int i, int j, int m)
{
    for (int k = 0; k <= m - 1; k++)
        HoanVi(a[i + k], a[j + k]);
}

void Transpose(int a[MAX], int n, int m)
{
    int i, j;
    i = m;
    j = n - m;
    m = m + 1;
```

```
while (i != j)
{
    if (i > j)
    {
        Exchange(a, m - i, m, j);
        i = i - j;
    }
    else
    {
        j = j - i;
        Exchange(a, m - i, m + j, i);
    }
    Exchange(a, m - i, m, i);
}
```

Bước 4:

Trong Program.cpp, cập nhật lại hàm ChayChuongTrinh() :

```
void ChayChuongTrinh()
{
    int a[MAX], n;
    int m;
    cout << "\nn = "; cin >> n;
    NhapDay(a, n);
    cout << "\nm = "; cin >> m;

    cout << "\nDay ban dau:\n";
    XuatDay(a, n);
    Transpose(a, n, m);
    cout << "\nDay ket qua:\n";
    XuatDay(a, n);
    cout << endl;
    system("PAUSE");
}
```

Chạy chương trình kiểm tra kết quả.

Kết thúc chương trình.

Bài 3.

Sử dụng thuật toán quay lui, viết chương trình tùy chọn thực hiện các thao tác :

- Liệt kê các dãy nhị phân độ dài n
- Liệt kê các tổ hợp chập k trong tập n số nguyên dương đầu tiên.
- Liệt kê các hoán vị

Bước 1 : Tạo project tên **Bai3_D_ql**, có các tập tin rỗng : **thuvien.h**, **menu.h**, **program.cpp**

Bước 2 :

Trong tập tin program.cpp, soạn phần code tối thiểu để chạy được chương trình nhằm kiểm tra các phần sẽ thêm vào chương trình.

```
#include <iostream>
using namespace std;
```

```
#include "Thuvien.h"
#include "Menu.h"
```

```
void ChayChuongTrinh();
int main()
{
    ChayChuongTrinh();
    return 1;
}
```

```
void ChayChuongTrinh()
{
}
```

Nhấn ctrl+F5 để chạy chương trình.

Bước 3 :

Viết các hàm tổ chức hệ thống menu trong thư viện menu.h :

//Hàm xuất menu

```
void XuatMenu()
```

```
{
    cout << "\n=====He thong menu =====";
    cout << "\n0. Thoat khoi chuong trinh";
    cout << "\n1. Nhap du lieu n";
    cout << "\n2. Xem du lieu n";
    cout << "\n3. Liet ke day nhi phan";
    cout << "\n4. Liet ke to hop ";
    cout << "\n5. Liet ke hoan vi";
}
```

//Hàm Chọn menu từ 0 đến soMenu

```
int ChonMenu(int soMenu)
```

```
{
    int stt;
    for (;;)
    {
        system("CLS");
        XuatMenu();
        cout << "\nNhap 1 so trong khoang [0,...," << soMenu
            << "] de chon menu (Lan dau tien chon 1),stt = ";
        cin >> stt;
        if (0 <= stt && stt <= soMenu)
            break;
    }
    return stt;
}
```

//Hàm xử lý menu

```
void XuLyMenu(int menu, int &n)
```

```
{
    switch (menu)
    {
        case 0 :
            system("CLS");
            cout << "\n0. Thoat khoi chuong trinh\n";
            break;
        case 1:
            system("CLS");
            cout << "\n1. Nhap du lieu";
            break;
        case 2:
            system("CLS");
            cout << "\n2. Xem du lieu";
            break;
        case 3:
            system("CLS");
            cout << "\n3. Liet ke day nhi phan";
            _getch();
            break;
        case 4:
            system("CLS");
    }
```



```
        cout << "\n4. Liet ke to hop";  
        break;  
    case 5:  
        system("CLS");  
        cout << "\n5. Liet ke hoan vi";  
        break;  
    }  
}
```

Viết xong mỗi hàm, nhấn ctrl+F5 để kiểm tra có lỗi hay không.

Bước 4 : vận hành hệ thống menu, chọn menu = 0 thì dừng chương trình
Trong tập tin program.cpp, viết lại hàm ChayChuongTrinh () như sau :

```
void ChayChuongTrinh()  
{  
    int menu, //luu trữ giá trị chọn stt của người dùng  
        soMenu = 5; //số chức năng trong chương trình  
    int n = 0; //khởi tạo kích thước dữ liệu  
    do  
    {  
        menu = ChonMenu(soMenu);  
        XuLyMenu(menu, n);  
        system("PAUSE"); //tạm dừng chương trình sau khi thực hiện một chức năng  
    } while (menu > 0); //menu = 0 thì dung chương trình  
}
```

Nhấn ctrl+F5 để chạy chương trình kiểm tra sửa lỗi và xem có vận hành đúng không.

Bước 5 : Nhập xuất dữ liệu

Trong trường hợp đơn giản này (chỉ nhập xuất số nguyên dương n) ta viết luôn trong hàm XuLyMenu() :

```
void XuLyMenu(int menu, int &n)  
{  
    switch (menu)  
    {  
    case 0 :  
        system("CLS");  
        cout << "\n0. Thoat khoi chuong trinh\n";  
        break;  
    case 1:  
        system("CLS");  
        cout << "\n1. Nhap du lieu";  
        cout << "\nn = ";  
        cin >> n;  
        break;  
    case 2:  
        system("CLS");  
        cout << "\n2. Xem du lieu";  
        cout << "\nn = " << n;  
        break;  
    case 3:  
        system("CLS");  
        cout << "\n3. Liet ke day nhi phan";  
        break;  
    case 4:  
        system("CLS");  
        cout << "\n4. Liet ke to hop";  
        break;  
    case 5:
```

```
        system("CLS");  
        cout << "\n5. Liet ke hoan vi";  
        break;  
    }  
}
```

Từ bước 6 trở về sau, trong tập tin thuvien.h ta viết code giải quyết các chức năng bài toán, xong chức năng nào ta thực hiện ngay (bổ sung vào hàm XuLyMenu)

Bước 6 :

Giải quyết bài toán liệt kê dãy nhị phân độ dài n.

a. Trong tập tin thuvien.h, ta soạn thảo :

```
#define MAX 20  
//Các biến toàn cục  
int n,  
    k;  
int a[MAX], //luu tru day nhi phan, hoan vi, to hop  
    b[MAX]; //danh dau  
int dem;    //dem day nhi phan, hoan vi, to hop
```

//Xuat day nhi phan, hoan vi, to hop

```
void Xuat_KQ(int a[MAX],int n)  
{  
    int i;  
    cout << "kq" << setw(3) << dem<< " : ";  
    for (i = 1; i <= n; i++)  
        cout << setw(2) << a[i];  
    cout << endl;  
}
```

//Day nhi phan chieu dai n

```
void LietKe_DayNP(int i)//Try  
{  
    int j;  
    for (j = 0; j <= 1; j++)  
    {  
        a[i] = j;  
        if (i < n)  
            LietKe_DayNP(i + 1);  
        else  
        {  
            dem++;  
            Xuat_KQ(a,n);  
        }  
    }  
}
```

b. Trong hàm XuLyMenu trong menu.h, ta bổ sung thực hiện vào case 3 như sau :

```
case 3:  
    system("CLS");  
    cout << "\n3. Liet ke day nhi phan";  
    cout << endl;  
    dem = 0;  
    LietKe_DayNP(1);  
    break;
```

Nhấn Ctrl+F5 để chạy chương trình, kiểm tra lỗi và kết quả.

Bước 7.

Giải quyết bài toán liệt kê tổ hợp chập k trong n

a. Trong tập tin thuvien.h, ta soạn thảo tiếp :

```
//Bai toan liet ke to hop
//To hop chap k trong n
void LietKe_TH(int i)//Try
{
    int j;
    for (j = a[i - 1] + 1; j <= n - k + i; j++)
    {
        a[i] = j;
        if (i == k)
        {
            dem++;
            Xuat_KQ(a,k);
        }
        else
            LietKe_TH(i + 1);
    }
}
```

b. Trong hàm XuLyMenu trong menu.h, ta bổ sung thực hiện vào case 4 như sau :

```
case 4:
    system("CLS");
    cout << "\n4. Liet ke to hop:\n";
    a[0] = 0;
    dem = 0;
    cout << "\nNhap k = ";
    cin >> k;
    LietKe_TH(1);
    break;
```

Chạy chương trình kiểm tra lỗi và kết quả.

Bước 8.

Giải quyết bài toán liệt kê các hoán vị n số nguyên dương đầu tiên

a. Trong tập tin thuvien.h, ta soạn thảo tiếp :

```
//Khoi tao hv
void KhoiTao_danh dau()
{
    int i;
    for (i = 1; i <= n; i++)
        b[i] = 1;
}
//Hoan vi n so nguyen duong dau tien
void LietKe_HV(int i)//Try
{
    int j;
    for (j = 1; j <= n; j++)
        if (b[j])
        {
            a[i] = j;
            b[j] = 0;
            if (i == n)
            {
                dem++;
                Xuat_KQ(a,n);
            }
            else
                LietKe_HV(i + 1);
            b[j] = 1;
        }
}
```

b. Trong hàm XuLyMenu trong menu.h, ta bổ sung thực hiện vào case 5 như sau :

case 5:

```
system("CLS");
cout << "\n5. Liet ke hoan vi\n";
dem = 0;
KhoiTao_danh dau();
LietKe_HV(1);
break;
```

Chạy chương trình kiểm tra lỗi và kết quả.

Kết thúc chương trình.

Ghi chú :

Trong “ThuVien.h” , sau khi viết xong mỗi hàm, ta có thể khai báo nguyên mẫu các hàm vào đầu tập tin (phía dưới các khai báo báo toàn cục)

Bài 4. Bài toán Ngựa đi tuần.

Cho bàn cờ có $n \times n$ ô . Một con ngựa được phép đi theo luật cờ vua, đầu tiên được đặt ở ô có tọa độ x_0, y_0 .

Vấn đề là hãy chỉ ra tất cả hành trình (nếu có) của ngựa – Đó là ngựa đi qua tất cả các ô của bàn cờ, mỗi ô đi qua đúng một lần .

Bước 1 : Tạo project đặt tên : Bai4_D_ql_MDT gồm các tập tin rỗng : thuvien.h và program.cpp.

Bước 2 :

Trong tập tin program.cpp, soạn phần code tối thiểu để chạy được chương trình nhằm kiểm tra các phần sẽ thêm vào chương trình.

```
#include <iostream>
#include <iomanip>
using namespace std;
#include "Thuvien.h"
void ChayChuongTrinh();
int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    system("PAUSE");
}
```

Bước 3 : Trong tập tin thuvien.h, ta soạn phần code thực hiện giải quyết bài toán.

Để dễ xử lý lỗi, khi viết thêm một hàm nào thì ta chạy ngay chương trình để kiểm tra lỗi.

- Nội dung tập tin thuvien.h :

```
//Khai bao hang
#define MAX 10

//Khoi dau cac buoc di cua ngua co the co
int a[8] = { 2, 1, -1, -2, -2, -1, 1, 2 };
int b[8] = { 1, 2, 2, 1, -1, -2, -2, -1 };
//Khai bao nguyen mau cac ham
void Try(int i, int x, int y, int h[MAX][MAX], int n, int &slg);
void Xuat_HanhTrinh(int h[MAX][MAX], int n);
void Init(int h[MAX][MAX], int n);

//*****
//Thu tuc de quy ngua thu tu o co toa do (x,y) di buoc ke tiep
```

```
//Xuat hanh trinh neu di het cac o
void Try(int i, int x, int y, int h[MAX][MAX], int n, int &slg)
{
    int k, u, v;
    for (k = 0; k <= 7; k++)
    {
        u = x + a[k];
        v = y + b[k];
        if (1 <= u && u <= n && 1 <= v && v <= n && h[u][v] == 0)
        {
            h[u][v] = i;
            if (i < n*n)
                Try(i + 1, u, v, h, n, slg);
            else
            {
                slg++;
                cout << "\nLoi giai thu " << slg << " :";
                Xuat_HanhTrinh(h, n);
            }
            h[u][v] = 0;
        }
    }
}

// *****
//Xua ma tran h : hanh trinh cua ngua
void Xuat_HanhTrinh(int h[MAX][MAX], int n)
{
    int i, j;
    cout << setiosflags(ios::left);
    for (i = 1; i <= n; i++)
    {
        cout << endl << endl;
        for (j = 1; j <= n; j++)
        {
            cout << setw(5) << h[i][j];
        }
        cout << endl << endl;
    }
}

// *****
//Khoi tao h
void Init(int h[MAX][MAX], int n)
{
    int i, j;
    for (i = 1; i <= n; i++)
        for (j = 1; j <= n; j++)
            h[i][j] = 0;
}

// *****
```

Bước 4 : Cập nhật hàm ChayChuongTrinh() trong program.cpp để thực hiện yêu cầu bài toán:

```
void ChayChuongTrinh()
{
    int n; //Kich thước ban co
    int x, y; //toa do dau
    int h[MAX][MAX]; //ma tran luu duong di cua ngua
    int slg = 0;
```

```

system("CLS");
cout << endl << "Nhập kích thước bàn cờ: n = ";
cin >> n;
cout << "\nNhập tọa độ đầu ";
cout << "\nx = ";
cin >> x;
cout << "\ny = ";
cin >> y;
system("CLS");
Init(h,n);
h[x][y] = 1;
Try(2, x, y,h,n,slg);
if (!slg)
    cout << "\nKhông có lời giải!";
else
    cout << "\nSố lời giải : " << slg;
system("PAUSE");
}

```

E. Bài tập

Bài 1.

Chia m phần thưởng cho n học sinh có thứ tự.

Bài 2

Lập lịch thi đấu vòng tròn 1 lượt cho n đội bóng đá, n là lũy thừa 2. Trong 1 đợt thi đấu, mỗi đội đấu 1 trận. Đấu trong n-1 đợt.

Kỹ thuật chia để trị xây dựng lịch cho một nửa số đội. Lịch này được lập nên do áp dụng đệ quy của thuật toán bằng cách tìm lịch cho một nửa số đội ... khi chỉ còn 2 đội thì ta có một cặp đấu đơn giản.

Các đội được đánh số từ 1 đến n. Giả sử có 8 đội. Lịch thi đấu cho 4 đội từ 1 đến 4 lấp đầy góc trái trên (4 hàng 3 cột) coi như đã lập xong. Góc trái dưới (4 hàng 3 cột) phải cho vào các đội có số thứ tự cao (từ 5 đến 8) ngược với các số khác. Lịch con này tạo ra bằng cách cộng 4 cho mỗi số nguyên của góc trái trên. Bây giờ ta đã làm đơn giản bài toán. Tất cả phần còn lại là các đội có số thấp đấu với các đội có số cao. Điều đó dễ hiểu là các đội từ 1-4 đấu với các đội 5-8 tương ứng từ đợt thứ 4 và hoán vị theo chu kỳ 5 đến 8 trong các đợt tiếp theo.

	đợt	1		đợt	1	2	3		đợt	1	2	3	4	5	6	7
đội	1	2		1	2	3	4		1	2	3	4	5	6	7	8
	2	1		2	1	4	3		2	1	4	3	6	7	8	5
				3	4	1	2		3	4	1	2	7	8	5	6
	4	3		4	3	2	1		4	3	2	1	8	5	6	7
									5	6	7	8	1	4	3	2
									6	5	8	7	2	1	4	3
									7	8	5	6	3	2	1	4
									8	7	6	5	4	3	2	1

Bài 3.

Cho một ma trận vuông cấp n, các phần tử của ma trận là các số tự nhiên. Ta nói đường đi trong ma trận là một đường gấp khúc không tự cắt xuất phát từ một ô nào đó của ma trận, sau đó có thể đi theo các hướng: lên trên, xuống dưới, rẽ trái, rẽ phải. Độ dài của đường đi là số ô nằm trong đường đi. Trọng số của đường đi là tổng các giá trị của các ô trên đường đi.

Với ô xuất phát cho trước, liệt kê tất cả các đường đi có thể, độ dài và trọng số của nó.

Bài 4.

Cho bàn cờ n x n ô. Mỗi ô được gán một số nguyên dương. Một con ngựa được phép đi theo luật cờ vua.

Xét các hành trình của ngựa, bắt đầu từ ô $\langle x_0, y_0 \rangle$, ô kết thúc $\langle x_1, y_1 \rangle$, mỗi ô trong hành trình ngựa đi qua đúng một lần.

Dùng thuật giải quay lui:

1. Liệt kê các hành trình của ngựa.
2. Ta nói giá trị của hành trình là tổng các số mỗi ô ngựa đi qua trong hành trình. Hãy chỉ ra hành trình có giá trị lớn nhất.

Bài 5.

Bài toán 8 hậu “Đặt trên bàn cờ vua 8 quân cờ hậu sao cho chúng không ăn được nhau”

LAB 02. Thuật giải tìm kiếm trong

A. Mục tiêu

Tìm hiểu và cài đặt các thuật giải tìm kiếm trong

B. Yêu cầu

- Sinh viên tự ôn tập (mục C)

- Buổi thực hành 2 (3 t) :

- Sinh viên thực tập trong phòng lab :
 - **Thời gian : 2 tiết đầu**
 - Khối lượng và nội dung : Các bài phần D (luyện tập)
 - Yêu cầu lưu trữ
 - ✓ Tạo thư mục **MaSV_HoVaTen_Lab02** chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
- Sinh viên tiếp tục thực tập trong phòng lab :
 - **Thời gian : 1 tiết (cuối)**
 - Khối lượng và nội dung : Bài 1 E (bài tập)
 - Yêu cầu lưu trữ
 - ✓ Tạo project cho bài.
 - ✓ Xóa thư mục debug trong project
 - ✓ Lưu trữ trong thư mục trên
 - ✓ Nén thư mục
 - Hình thức thu bài
 - ✓ Hệ thống thu bài trong phòng lab.

C. Ôn tập

1. Thuật giải tìm kiếm tuyến tính

a. Phát biểu bài toán :

Tìm x có trong dãy a?

- Input : $a_0, a_2, \dots, a_{n-1}, x$
int a[n], x;
- Output :
 - Nếu có, trả về 1 ;
 - Nếu không có, trả về 0

b. Mô tả thuật giải:

- Bước 1: Xuất phát từ phần tử đầu tiên của dãy: **i=0**
- Bước 2: So sánh a[i] với giá trị x, có 2 trường hợp:
 - $a[i] = x$: tìm thấy, dừng thuật giải
 - $a[i] \neq x$: sang bước 3
- Bước 3: Xét phần tử kế tiếp trong mảng: **i = i+1**
 - nếu $i > n-1$: hết mảng, không tìm thấy, dừng thuật giải
 - ngược lại: quay lại bước 2.

b. Cài đặt:

- TH1: Không dùng lính canh

int LinearSearch (int a[], int n, int x)

```
{  
    int i = 0;  
    while ((i < n) && (a[i] != x)  
        i++;  
    if (i == n)  
        return 0; // tìm hết mảng nhưng không có x  
    return 1; // tìm thấy x tại vị trí x  
}
```

}

- TH2: Dùng lính canh

- Đặt thêm phần tử có giá trị x vào cuối mảng (luôn tìm thấy x trong mảng)
- Dựa vào vị trí tìm thấy x để kết luận.

int LinearSearch (int a[], int n, int x)

```
{
    int i = 0;
    a[n] = x; // đặt phần tử lính canh
    while (a[i] != x)
        i++;
    if (i == n)
        return 0; // tìm hết mảng nhưng không có x
    return 1; // tìm thấy x
}
```

2. Thuật giải tìm kiếm nhị phân

(Chỉ sử dụng cho các dãy đã có thứ tự, giả sử là dãy tăng)

a. Phát biểu bài toán :

Tìm x có trong dãy tăng a?

- Input : $a_0, a_2, \dots, a_{n-1}, x$
int a[n], x;
- Output :
 - Nếu có, trả về 1 ;
 - Nếu không có, trả về 0

b. Ý tưởng (chia để trị) :

Giả sử dãy đã có thứ tự tăng : $i < j \Rightarrow a_i \leq a_j$

- Nếu $x > a_k$ thì x chỉ có thể xuất hiện trong đoạn $[a_{k+1}, a_{n-1}]$
- Nếu $x < a_k$ thì x chỉ có thể xuất hiện trong đoạn $[a_0, a_{k-1}]$

c. Mô tả thuật giải:

- Bước 1: left=0; right = n-1; // tìm trên tất cả các phần tử
- Bước 2: mid = (left+right)/2; // lấy mốc so sánh
So sánh a[mid] với giá trị x, có 3 trường hợp:
 - $a[mid] = x$: tìm thấy, trả về 1, dừng thuật giải
 - $a[mid] > x$: right = mid-1 // tìm tiếp trong dãy con $a_{left} \dots a_{mid-1}$
 - $a[mid] < x$: left = mid+1 // tìm tiếp trong dãy con $a_{mid+1} \dots a_{right}$
- Bước 3:
 - nếu left ≤ right: lặp lại bước 2 // còn phần tử chưa xét, tìm tiếp
 - ngược lại: trả về 0, dừng // đã xét hết mọi phần tử

d. Cài đặt:

int BinarySearch (int a[], int n, int x)

```
{
    int left = 0, right = n-1, mid;
    do
    {
        mid = (left+right)/2;
        if (x == a[mid])
            return 1; // tìm thấy x tại vị trí mid
        else
            if (x < a[mid])
                right = mid - 1;
            else
                left = mid + 1;
    }
    while (left <= right);
    return 0; // tìm hết dãy mà không có x
}
```

Có thể cài đặt đệ quy như sau :


```
int BinarySearch_R (int a[],int x,int l, int r)
{
    int mid;
    if ( l > r )
        return 0;
    mid = (l+r)/2;
    if ( x == a[mid] )
        return 1;
    else
        if ( x > a[mid] )
            return BinarySearch_R (a,x,mid+1,r);
        else
            return BinarySearch_R (a,x,l,mid-1);
}
```

3. Thuật giải tìm kiếm nội suy

Hoạt động như tìm kiếm nhị phân, chỉ khác việc chọn chỉ số mid như sau :

$$mid = left + \frac{x - a[left]}{a[right] - a[left]} * (right - left);$$

- Cài đặt thuật toán :

```
int TimNoiSuy(int a[MAX], int n, int x)
{
    int left = 0, right = n - 1, mid;
    while (left <= right)
    {
        mid = left + (right - left) * (x - a[left]) / (a[right] - a[left]);

        if (a[mid] == x)
            return 1;
        else
            if (a[mid] > x)
                right = mid - 1;
            else
                left = mid + 1;
    }
    return 0; // không tìm thấy
}
```

D. LUYỆN TẬP

Bài 1:

Tìm kiếm số nguyên x trên dãy n số nguyên a[0..n-1]:

- Nếu không có, trả về -1;
- Nếu có, trả về các trường hợp sau:
 - Chỉ số đầu tiên
 - Chỉ số cuối cùng
 - Chỉ số của các phần tử trong dãy trùng x.

Sử dụng thuật giải tìm kiếm tuyến tính.

- Dãy số được tạo ra từ tập tin test1.txt :

```
9 //Kích thước mảng
0      -1      0      6      5      0      5      0      3
```

Khóa x : 0, 1, 5

- Dãy số được tạo ra từ tập tin test2.txt :

```
9 //Kích thước mảng
3      8      9      9      0      9      0      5      9
```

Khóa x : 0, 1, 9

Bước 1.

Trong thư mục **MaSV_HoVaTen_Lab02**, tạo Project với tên là Lab02_Bai1D gồm các tập tin **Thuvien.h**, **Menu.h**, **Program.cpp**

Bước 2.

- Soạn thảo nội dung tập tin Program.cpp (phần code tối thiểu chạy chương trình)

```
#include <iostream>
#include <fstream>

using namespace std;

#include "Thuvien.h"
#include "Menu.h"

void ChayChuongTrinh();

int main()
{
    ChayChuongTrinh();
    return 1;
}

void ChayChuongTrinh()
{
    system("PAUSE");
}
```

Bước 3. Tạo hệ thống tùy chọn menu

- Trong tập tin **ThuVien.h** :

// Định nghĩa hằng

#define MAX 100 //kích thước khai báo mảng

- Trong tập tin **menu.h**, soạn thảo các hàm tổ chức menu :

//Hàm xuất các tên chức năng của chương trình

```
void XuatMenu()
{
    cout << "\n===== He thong chuc nang =====";
    cout << "\n0. Thoat khoi chuong trinh";
    cout << "\n1. Tao du lieu";
    cout << "\n2. Xem du lieu";
    cout << "\n3. Tim kiem tuyen tinh - Tra ve chi so dau tien";
    cout << "\n4. Tim kiem tuyen tinh - Tra ve chi so i dau tien, neu co(Co linh canh)";
    cout << "\n5. Tim kiem tuyen tinh - Tra ve chi so cuoi cung, neu co";
    cout << "\n6. Tra ve tat ca chi so i neu co.";
}

//Hàm điều khiển người dùng chọn chỉ số chức năng hợp lệ : từ 0 đến soMenu
int ChonMenu(int soMenu)
{
    int stt;
    for (;;)
    {
        system("CLS"); //xoa man hinh
        XuatMenu();
        cout << "\nNhap mot so ( 0 <= so <= " << soMenu << " ) de chon menu, stt = ";
        cin >> stt;
        if (0 <= stt && stt <= soMenu)
            break;
    }
    return stt;
}
```

```
}

//Hàm xử lý menu
void XuLyMenu(int menu, int a[MAX], int &n)
{
    switch (menu)
    {
        case 0:
            cout << "\n0. Thoat khoi chuong trinh\n";
            break;

        case 1:
            cout << "\n1. Tao du lieu";
            cout << "\nNhap ten tap tin, filename = ";
            break;

        case 2:
            cout << "\n2. Xem du lieu";
            break;

        //...
    }
}
```

- Khai báo nguyên mẫu vào đầu tập tin menu.h :

```
void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, int a[MAX], int &n);
```

Bước 4. Vận hành menu của chương trình

- Trong hàm ChayChuongTrinh() của tập tin Program.cpp, ta cập nhật lại như sau :

```
void ChayChuongTrinh()
{
    int    soMenu = 6,
           menu;
    int    a[MAX],
           n = 0;
    do
    {
        system("CLS");
        menu = ChonMenu(soMenu);
        XuLyMenu(menu, a, n);
        system("PAUSE");

    } while (menu > 0);
}
```

Bước 5 : Tạo dữ liệu, xuất dữ liệu, tạo tập tin dữ liệu; sử dụng các hàm chức năng

- Trong tập tin thuvien.h, viết các hàm tạo và xuất dữ liệu

1. Hàm tạo dữ liệu

//Hàm chuyển dữ liệu từ tập tin văn bản vào mảng 1 chiều

```
int TapTin_mang1c(char *filename, int a[MAX], int &n)
{
    int i;
    ifstream in(filename);    //Mở tập tin filename để đọc
    if (!in)
        return 0; //không thành công
    in >> n;
    for (i = 0; i < n; i++)
        in >> a[i];
```

```
in.close();
return 1;//thanh cong
}
```

2. Hàm xuất dữ liệu :

```
void Xuat_Mang(int a[MAX], int n)
{
    for (int i = 0; i < n; i++)
        cout << a[i] << '\t';
}
```

- Tạo các tập tin dữ liệu theo yêu cầu : test1.txt, test2.txt

+ Nhấn tổ hợp phím chuột lên Resource Files, chọn add, chọn new item. Tiếp theo, trong Visual C++ chọn Utility, chọn test file (.txt), trong name đặt tên test1.txt, soạn nội dung theo yêu cầu của tập tin như sau :

```
9
0      -1      0      6      5      0      5      0      3
```

+ Tương tự cho test2.txt:

```
10
3      8      9      9      0      9      0      5      9      9
```

- Sử dụng các hàm tạo dữ liệu , xuất dữ liệu trong hàm XuLyMenu của menu.h như sau :

```
void XuLyMenu(int menu, int a[MAX], int &n)
{
    int kq;
    char filename[MAX];
    switch (menu)
    {
        case 0:
            system("CLS");
            cout << "\n0. Thoat khoi chuong trinh\n";
            break;
        case 1:
            system("CLS");
            cout << "\n1. Tao du lieu";
            do
            {
                cout << "\nNhap ten tap tin, filename = ";
                cin >> filename;
                kq = TapTin_mang1c(filename, a, n);
            }while(!kq);
            cout << "\nMang vua tao:\n";
            Xuat_Mang(a, n);
            cout << endl;
            break;
        case 2:
            system("CLS");
            cout << "\n2. Xuat du lieu";
            cout << "\nMang vua tao:\n";
            Xuat_Mang(a, n);
            cout << endl;
            break;
        //...
    }
}
```

Bước 6 (và các bước kế tiếp cho đến hết các chức năng bài toán)

- Soạn thảo từng hàm chức năng trong thuvien.h

- Sử dụng hàm chức năng vào case tương ứng trong hàm void XuLyMenu(int menu, int a[MAX], int &n)

- Tham khảo tập tin **thuvien.h** đã bổ sung các hàm chức năng còn lại :

//Tra ve chỉ số i đầu tiên, nếu có x

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

//Co linh canh, tra ve chỉ số i đầu tiên, nếu có x

```
int TKTT_DauTien_LC(int a[MAX], int n, int x)
{
    int i = 0;
    a[n] = x; //Linh canh
    while (a[i] != x) i++;
    if (i == n)
        return -1;
    return i;
}
```

//Tra ve chỉ số i cuối cùng, nếu có x

```
int TKTT_CuoiCung(int a[MAX], int n, int x)
{
    int i = n - 1;
    while ((i >= 0) && (a[i] != x)) i--;
    return i;
}
```

//Tra ve các chỉ số xuất hiện, nếu có x

```
void TKTT_CacChiSo(int a[MAX], int n, int x)
{
    int i, kq = -1;
    for (i = 0; i < n; i++)
        if (a[i] == x)
        {
            kq = 1;
            break;
        }
    if (kq == -1)
        cout << endl << x << " không có trong a";
    else
    {
        cout << endl << x << " xuất hiện trong a tại các vị trí :\n";
        for (i = 0; i < n; i++)
            if (a[i] == x)
                cout << i << "t";
    }
}
```

Lưu ý : Khai báo nguyên mẫu các hàm vào vùng khai báo ở đầu tập tin:

```
int TapTin_mang1c(char *filename, int a[MAX], int &n);
void Xuat_Mang(int a[MAX], int n);
int TKTT_DauTien(int a[MAX], int n, int x);
int TKTT_DauTien_LC(int a[MAX], int n, int x);
int TKTT_CuoiCung(int a[MAX], int n, int x);
```

```
void TKTT_CacChiSo(int a[MAX], int n, int x);
```

- Tham khảo hàm XuLyMenu trong tập tin *menu.h* đã bổ sung các nội dung còn lại :

```
void XuLyMenu(int menu, int a[MAX], int &n)
{
    int kq;
    int x;
    char filename[MAX];
    switch (menu)
    {
        //...
    case 3:
        cout << "\n3. Timkiem tuyen tinh - Tra ve chi so dau tien";
        cout << "\nMang du lieu ban dau:\n";
        Xuat_Mang(a, n);
        cout << "\nNhap x = ";
        cin >> x;
        kq = TKTT_DauTien(a, n, x);
        if (kq == -1)
            cout << endl << x << " khong co trong mang";
        else
            cout << endl << x << " xuất hiện trong a tại vị trí đầu tiên là : " << kq;
        cout << endl;
        break;
    case 4:
        cout << "\n4. Timkiem tuyen tinh (co linh canh)- Tra ve chi so dau tien";
        cout << "\nMang du lieu ban dau:\n";
        Xuat_Mang(a, n);
        cout << "\nNhap x = ";
        cin >> x;
        kq = TKTT_DauTien_LC(a, n, x);
        if (kq == -1)
            cout << endl << x << " khong co trong mang";
        else
            cout << endl << x << " xuất hiện trong a tại vị trí đầu tiên là : " << kq;
        cout << endl;
        break;
    case 5:
        cout << "\n5. Timkiem tuyen tinh - Tra ve chi so cuoi cung";
        cout << "\nMang du lieu ban dau:\n";
        Xuat_Mang(a, n);
        cout << "\nNhap x = ";
        cin >> x;
        kq = TKTT_CuoiCung(a, n, x);
        if (kq == -1)
            cout << endl << x << " khong co trong mang";
        else
            cout << endl << x << " xuất hiện trong a tại vị trí cuối cùng là : " << kq;
        cout << endl;
        break;
    case 6:
        cout << "\n6. Tra ve tat ca chi so i neu co.";
        cout << "\nMang du lieu ban dau:\n";
        Xuat_Mang(a, n);
        cout << "\nNhap x = ";
        cin >> x;
        TKTT_CacChiSo(a, n, x);
        cout << endl;
        break;
    }
```

```
}
}
```

Bài 2:

Giả sử có một danh sách sinh viên, mỗi một sinh viên được lưu trữ các thông tin:

- Mã sinh viên, //Một chuỗi có 7 ký tự, không có ký tự trắng
- Họ của sinh viên, //một chuỗi có không quá 9 ký tự
- Tên lót, // một chuỗi có không quá 9 ký tự
- Tên sinh viên, // một chuỗi có không quá 9 ký tự
- Năm sinh, số nguyên dương
- Lớp, // một chuỗi có 5 ký tự, không có ký tự trắng
- Điểm trung bình, //Một số thực từ 0 đến 10
- Tổng số tín chỉ đã tích lũy được, một số nguyên từ 0 đến 50

Thực hiện các thao tác tìm kiếm trên danh sách sinh viên:

- Tìm kiếm theo mã sinh viên .
- Tìm kiếm theo tên: Xuất tất cả các sinh viên trùng tên.
- Tìm kiếm theo họ: Xuất tất cả các sinh viên trùng họ.
- Tìm kiếm theo điểm trung bình : Xuất tất cả sinh viên có điểm trung bình $\geq x$.
- Tìm kiếm theo lớp : Xuất sinh viên thuộc lớp cho trước.
- Tìm kiếm nhĩ phân theo trường Tích lũy (nếu trường Tích lũy có thực tự)

Các chuỗi lưu trữ thông tin về họ, tên lót, tên có thể gồm nhiều từ, các từ được nối với nhau bởi dấu gạch dưới. Nếu tên lót không có thì dùng dấu gạch dưới thay thế.

Danh sách sinh viên được cho trong tập tin văn bản **test1.txt** như sau :

```

=====
:Ma SV      :      Ho va Ten sinh vien      :Lop      :NS      :DTB      :TichLuy      :
=====
:1412045    :Nguyen      Tuan      Uo      :CTK38     :1996     :6.2      :30      :
:1443210    :Truong      Thi      Hoa      :CTK38     :1995     :7.3      :31      :
:1423452    :Tran        Ngoc      Ninh     :CTK36     :1994     :8.6      :32      :
:1334432    :Hoang       Hoa      :CTK37     :1995     :5.2      :36      :
:1342332    :Le          Thi      Lieu     :CTK37     :1994     :4.6      :37      :
:1342032    :Van         Thi      Hoa      :CTK38     :1994     :6.6      :38      :
:1223052    :Uo          Ngoc     Hoa      :CTK36     :1994     :6.6      :40      :
:1312040    :Nguyen      Van      Uo        :CTK37     :1996     :6.2      :41      :
:1212045    :Nguyen      Thi      Lieu     :CTK37     :1995     :9.2      :42      :
:1313045    :Tran        Trong    Hieu     :CTK37     :1993     :4.1      :43      :
:1212042    :Ly          Van      Hoa      :CTK36     :1993     :5.3      :44      :
:1300432    :Le          -        Uo        :CTK37     :1995     :5.5      :45      :
=====

```

Lưu ý là trong cột " Ho va ten sinh vien " chính là bao gồm 3 cột "Họ sinh viên", "tên lót sinh viên", và "tên sinh viên".

Thực hiện chương trình bao gồm các bước sau đây :

Bước 1 :

Trong thư mục **MaSV_HoVaTen_Lab02** , tạo Project chứa với tên là Lab02_ Bai2D gồm các tập tin **Thuvien.h**, **Menu.h**, **Program.cpp**

Bước 2 :

- Trong tập tin Program.cpp soạn nội dung sau (code tối thiểu để chạy được chương trình)

```

#include <iostream>
#include <fstream> //do có nhập tập tin
#include <iomanip>
#include <string.h>
using namespace std;

```

```

#include "Thuvien.h"
#include "Menu.h"

```

```
void ChayChuongTrinh();
```

```
int main()
```

```
{  
    ChayChuongTrinh();  
    return 1;  
}
```

```
void ChayChuongTrinh()  
{  
    system("PAUSE");  
}
```

Nhấn Ctrl+F5 để chạy chương trình kiểm tra lỗi

Bước 3:

Trong tập tin "*ThuVien.h*" ta định nghĩa hằng, kiểu dữ liệu mới.

```
#define MAX 100  
struct sinhvien  
{  
    char    maSV[8];  
    char    hoSV[10];  
    char    tenLot[10];  
    char    ten[10];  
    char    lop[6];  
    int     namSinh;  
    double  dtb;  
    int     tinhLuy;  
};
```

Nhấn Ctrl+F5 chạy chương trình để kiểm tra lỗi

Bước 4 : Soạn thảo hệ thống tùy chọn menu.

Trong tập tin *menu.h*, ta soạn thảo các hàm sau :

```
//Hàm xuất menu  
void XuatMenu()  
{  
    cout << "\n===== He thong chuc nang =====";  
    cout << "\n0. Thoat khoi chuong trinh";  
    cout << "\n1. Tao danh sach sinh vien ";  
    cout << "\n2. Xem danh sach sinh vien";  
    cout << "\n3. Tim kiem theo ma sinh vien";  
    cout << "\n4. Tim kiem theo ten-Xuat cac sinh vien cung ten";  
    cout << "\n5. Tim kiem theo ho-Xuat cac sinh vien cung ho";  
    cout << "\n6. Xuat sinh vien co diem trung binh >= dtb cho truo";  
    cout << "\n7. Tim kiem theo lop-Xuat danh sach sinh vien trong lop";  
    cout << "\n8. Tim kiem nhi phan theo tinh luy (neu truong tinh luy co thu tu)";  
}
```

Nhấn Ctrl+F5 chạy chương trình để kiểm tra lỗi

```
//Hàm ChonMenu  
int ChonMenu(int soMenu)  
{  
    int stt;  
    for (;;)   
    {  
        system("CLS"); //xoa man hinh  
        XuatMenu();  
        cout << "\nNhap mot so (0 <= so <= " << soMenu << " ) de chon menu, stt = ";  
        cin >> stt;  
        if (0 <= stt && stt <= soMenu)  
            break;  
    }  
    return stt;  
}
```


}

Nhấn Ctrl+F5 chạy chương trình để kiểm tra lỗi

//Hàm Xử lý menu

```
void XuLyMenu(int menu, sinhvien a[MAX], int &n)
{
    switch (menu)
    {
        case 0:
            system("CLS");
            cout << "\n0. Thoat khoi chuong trinh\n";
            break;
        case 1:
            system("CLS");
            cout << "\n1. Tao Danh sach sinh vien.\n";
            break;
        case 2:
            system("CLS");
            cout << "\n2. Xem Danh sach sinh vien.\n";
            break;
        case 3:
            system("CLS");
            cout << "\n3. Tim kiem theo ma sinh vien";
            break;
        case 4:
            system("CLS");
            cout << "\n4. Tim kiem theo ten--Xuat cac sinh vien cung ten\n";
            break;
        case 5:
            system("CLS");
            cout << "\n5. Tim kiem theo ho--Xuat cac sinh vien cung ho\n";
            break;
        case 6:
            system("CLS");
            cout << "\n6. Xuat sinh vien co diem trung binh >= dtb cho truooc\n";
            break;
        case 7:
            system("CLS");
            cout << "\n7. Tim kiem theo lop--Xuat cac sinh vien thuoc lop\n";
            break;
        case 8:
            system("CLS");
            cout << "\n8. Tim kien nhi phan theo tich luy";
            break;
    }
}
```

Nhấn Ctrl+F5 chạy chương trình để kiểm tra lỗi

Bước 5 :

Khai báo nguyên mẫu các hàm tổ chức menu đầu tập tin **menu.h** :

//Khai bao nguyen mau

void XuatMenu();

int ChonMenu(int soMenu);

void XuLyMenu(int menu, sinhvien a[MAX], int &n);

Bước 6: Vận hành menu

- Chọn từ 1 đến soMenu thì chưa làm gì cả.

- Chọn 0 thì dừng được chương trình.

Viết lại hàm ChayChuongTrinh() trong tập tin **Program.cpp** như sau :

```
void ChayChuongTrinh()
{
    sinhvien a[MAX];
    int soMenu = 9,
        menu,
        n = 0; //so luong sinh vien trong danh sach
    //Vận hành hệ thống menu
    do
    {
        system("CLS");
        menu = ChonMenu(soMenu);
        XuLyMenu(menu, a, n);
        system("PAUSE");
    } while (menu > 0);
}
```

Nhấn Ctrl+F5 chạy chương trình để kiểm tra lỗi

Bước 7 : Viết các hàm tạo danh sách, xuất danh sách, tạo tập tin test1.txt theo yêu cầu :

1. Hàm tạo danh sách sinh viên :

//Hàm trả về 0 nếu không thành công

//Hàm trả về 1 nếu thành công

```
int TapTin_MangCT(char *filename, sinhvien a[MAX], int &n)
```

```
{
    ifstream in(filename);
    if (!in)
        return 0;

    char    maSV[8];
    char    hoSV[10];
    char    tenLot[10];
    char    ten[10];
    char    lop[6];
    int     namSinh;
    double  dtb;
    int     tichLuy;

    n = 0;
    in >> maSV; strcpy_s(a[n].maSV, maSV);
    in >> hoSV; strcpy_s(a[n].hoSV, hoSV);
    in >> tenLot; strcpy_s(a[n].tenLot, tenLot);
    in >> ten; strcpy_s(a[n].ten, ten);
    in >> lop; strcpy_s(a[n].lop, lop);
    in >> namSinh; a[n].namSinh = namSinh;
    in >> dtb; a[n].dtb = dtb;
    in >> tichLuy; a[n].tichLuy = tichLuy;

    while (!in.eof())
    {
        n++;
        in >> maSV; strcpy_s(a[n].maSV, maSV);
        in >> hoSV; strcpy_s(a[n].hoSV, hoSV);
        in >> tenLot; strcpy_s(a[n].tenLot, tenLot);
        in >> ten; strcpy_s(a[n].ten, ten);
        in >> lop; strcpy_s(a[n].lop, lop);
        in >> namSinh; a[n].namSinh = namSinh;
        in >> dtb; a[n].dtb = dtb;
        in >> tichLuy; a[n].tichLuy = tichLuy;
    }
}
```

```
n++;
in.close();
return 1;
}
```

Nhấn Ctrl+F5 chạy chương trình để kiểm tra lỗi

2. Hàm xuất danh sách sinh viên

//Hàm Xuất tiêu đề (tên các trường dữ liệu)

```
void Tieude()
{
    int i;
    cout << "\n";
    cout << ':';
    for (i = 1; i <= 74; i++)
        cout << '=';
    cout << ':';
    cout << "\n";

    cout << setiosflags(ios::left);
    cout << ':';
    cout << setw(9) << "Ma SV"
        << ':';
        << setw(30) << "Ho va Ten sinh vien"
        << ':';
        << setw(10) << "Lop"
        << ':';
        << setw(6) << "NS"
        << ':';
        << setw(6) << "DTB"
        << ':';
        << setw(8) << "TichLuy"
        << ':';

    cout << "\n";
    cout << ':';
    for (i = 1; i <= 74; i++)
        cout << '=';
    cout << ':';
    cout << "\n";
}
```

Nhấn Ctrl+F5 chạy chương trình để kiểm tra lỗi.

Các cột " họ sinh viên", "tên lót sinh viên" và "Tên sinh viên" ta ghép lại thành 1 cột tiêu đề " Họ và tên sinh viên. Các cột tiêu đề tách bằng dấu ':', các hàng bao các tên tiêu đề và kết thúc ta dùng các đường ký hiệu '='.

- Hàm Xuất 1 sinh viên ra màn hình.

```
void Xuat_SV(sinhvien p)
{
    cout << ':';
    cout << setiosflags(ios::left)
        << setw(9) << p.maSV
        << ':';
        << setw(10) << p.hoSV
        << setw(10) << p.tenLot
        << setw(10) << p.ten
        << ':';
        << setw(10) << p.lop
        << ':';
        << setw(6) << p.namSinh
        << ':';
        << setw(6) << setiosflags(ios::fixed) << setprecision(2) << p.dtb
        << ':';
}
```

```
<< setw(8) << p.tichLuy
<< '.';
}
```

Nhấn Ctrl+F5 chạy chương trình để kiểm tra lỗi.

- Hàm xuất danh sách sinh viên

```
void Xuat_DSSV(sinhvien a[MAX], int n)
```

```
{
    int i;
    TieuDe();
    for (i = 0; i < n; i++)
    {
        Xuat_SV(a[i]);
        cout << '\n';
    }
    cout << '.';
    for (i = 1; i <= 74; i++)
        cout << '=';
    cout << '.';
    cout << "\n";
}
```

Nhấn Ctrl+F5 chạy chương trình để kiểm tra lỗi.

- Tạo tập tin test1.txt :

Nhấn trỏ phải chuột lên Resource Files, chọn add, chọn new item. Tiếp theo, trong Visual C++ chọn Utility, chọn test file (.txt), trong name đặt tên test1.txt, soạn nội dung theo yêu cầu của tập tin như sau :

1412045	Nguyen	Tuan	Vo	CTK38	1996	6.2	30
1443210	Truong	Thi	Hoa	CTK38	1995	7.3	31
1423452	Tran	Ngoc	Ninh	CTK36	1994	8.6	32
1334432	Hoang	-	Hoa	CTK37	1995	5.2	36
1342332	Le	Thi	Lieu	CTK37	1994	4.6	37
1342032	Van	Thi	Hoa	CTK38	1994	6.6	38
1223052	Vo	Ngoc	Hoa	CTK36	1994	6.6	40
1312040	Nguyen	Van	Vu	CTK37	1996	6.2	41
1212045	Nguyen	Thi	Lieu	CTK37	1995	9.2	42
1313045	Tran	Trong	Hieu	CTK37	1993	4.1	43
1212042	Ly	Van	Hoa	CTK36	1993	5.3	44
1300432	Le	-	Vo	CTK37	1995	5.5	45

- Sử dụng hàm tạo danh sách sinh viên trong case 1, xuất danh sách sinh viên trong case 2 trong hàm XuLyMenu như sau :

```
void XuLyMenu(int menu, sinhvien a[MAX], int &n)
{
    int kq;
    char filename[MAX];
    switch (menu)
    {
        case 0:
            system("CLS");
            cout << "\n0. Thoat khoi chuong trinh\n";
            break;
        case 1:
            system("CLS");
            cout << "\n1. Tao danh sach sinh vien";
            do
            {
                cout << "\nNhap ten tap tin, filename = ";
                cin >> filename;
                kq = TapTin_MangCT(filename, a, n);
            }
            while (kq != -1);
    }
}
```

```

    } while (!kq);
    cout << "\nDanh sach sinh vien vua nhap:\n";
    Xuat_DSSV(a, n);

    cout << endl;
    break;
case 2:
    system("CLS");
    cout << "\n2. Xem danh sach sinh vien\n";
    cout << "\nDanh sach sinh vien hien hanh:\n";
    Xuat_DSSV(a, n);
    cout << endl;
    break;
// ...
}
}

```

Bước 8 và các bước tiếp theo cho đến hết :

Viết xong 1 hàm chức năng trong *ThuVien.h*, ta sử dụng ngay trong hàm *XuLyMenu* với case tương ứng.

- Các hàm chức năng còn lại trong *ThuVien.h*:

```

//Tim theo ma so: tra ve chi so i ĐT sao cho a[i].maSV = maSV
int Tim_MaSo_DauTien(char maSV[10], sinhvien a[MAX], int n)
{
    int i = 0;
    while ((i < n) && (_stricmp(a[i].maSV, maSV)))
        i++;
    if (i == n)
        return -1;
    return i;
}

//Tim theo ten
void Tim_TheoTen(char ten[10], sinhvien a[MAX], int n)
{
    int i, kq = -1;
    for (i = 0; i < n; i++)
        if (_stricmp(a[i].ten, ten) == 0)
        {
            kq = 1;
            break;
        }
    if (kq == -1)
        cout << "\nDanh sach khong co ten sinh vien : " << ten;
    else
    {
        cout << "\nthong tin sinh vien trong danh sach co ten " << ten;
        cout << endl;
        TieuDe();
        for (i = 0; i < n; i++)
            if (_stricmp(a[i].ten, ten) == 0)
            {
                cout << endl;
                Xuat_SV(a[i]);
            }
    }
}

//Tim theo ten
void Tim_TheoHo(char hoSV[10], sinhvien a[MAX], int n)
{
    int i, kq = -1;

```

```
for (i = 0; i < n; i++)
if (_stricmp(a[i].hoSV, hoSV) == 0)
{
    kq = 1;
    break;
}
if (kq == -1)
    cout << "\nDanh sach khong co sinh vien mang ho : " << hoSV;
else
{
    cout << "\nCac sinh vien trong danh sach mang ho " << hoSV << " :\n";
    cout << endl;
    TieuDe();
    for (i = 0; i < n; i++)
    if (_stricmp(a[i].hoSV, hoSV) == 0)
    {
        cout << endl;
        Xuat_SV(a[i]);
    }
}
}
//Tim theo lop
void Tim_TheoLop(char lop[6], sinhvien a[MAX], int n)
{
    int i, kq = -1;
    for (i = 0; i < n; i++)
    if (_stricmp(a[i].lop, lop) == 0)
    {
        kq = 1;
        break;
    }
    if (kq == -1)
        cout << "\nKhong co lop : " << lop;
    else
    {
        cout << "\nCac sinh vien trong danh sach thuoc lop " << lop << " :\n";
        cout << endl;
        TieuDe();
        for (i = 0; i < n; i++)
        if (_stricmp(a[i].lop, lop) == 0)
        {
            cout << endl;
            Xuat_SV(a[i]);
        }
    }
}
//Tim theo diem trung binh : xuat sinh vien co diem >= dtb
void Tim_TheoDTB(double dtb, sinhvien a[MAX], int n)
{
    int i, kq = -1;
    for (i = 0; i < n; i++)
    if (a[i].dtb >= dtb)
    {
        kq = 1;
        break;
    }
    if (kq == -1)
        cout << "\nKhong co sinh vien nao co diem trung binh >= " << dtb;
    else
```

```
{
    cout << "\nCac sinh vien trong danh sach co diem trung bin >= " << dtb << " :\n";
    cout << endl;
    TieuDe();
    for (i = 0; i < n; i++)
        if (a[i].dtb >= dtb)
        {
            cout << endl;
            Xuat_SV(a[i]);
        }
    }
}

//Kiem tra day so nguyen tang, tang : 1, khong tang : 0
int KiemTraDayTang(int x[MAX], int n)
{
    int i, kq = 1;
    for (i = 0; i < n - 1; i++)
        if (x[i] > x[i + 1])
        {
            kq = 0; //khong tang
            break;
        }
    return kq;
}

//Kiem tra day so nguyen giam, giam : 1, khong giam : 0
int KiemTraDayGiam(int x[MAX], int n)
{
    int i, kq = 1;
    for (i = 0; i < n - 1; i++)
        if (x[i] < x[i + 1])
        {
            kq = 0; //khong giam
            break;
        }
    return kq;
}

void TKNP_Theo_TichLuy(sinhvien a[MAX], int n)
{
    int i, kq;
    int x[MAX];
    for (i = 0; i < n; i++)
        x[i] = a[i].tichLuy;

    if (!KiemTraDayGiam(x, n) && !KiemTraDayTang(x, n))
    {
        cout << "\nDay so nguyen tao boi truong tich luy khong don dieu";
        cout << "\nKhong su dung duoc thuat giai tim kiem nhi phan!\n";

        return;
    }
    int tichLuy;
    cout << "\nNhap so tich luy : ";
    cin >> tichLuy;
    if (KiemTraDayTang(x, n))
    {
        kq = TKNP_Tang(x, n, tichLuy);
        Xuat_TKNP_Theo_TichLuy(tichLuy, a, n, kq);
    }
}
```

```
if (KiemTraDayGiam(x, n))
{
    kq = TKNP_Giam(x, n, tichLuy);
    Xuat_TKNP_Theo_TichLuy(tichLuy, a, n, kq);
}
}
int TKNP_Tang(int x[MAX], int n, int tichLuy)
{
    int kq = -1, midle, left = 0, right = n-1;
    do
    {
        midle = (left + right) / 2;
        if (tichLuy == x[midle])
        {
            kq = midle;
            break;
        }
        else
        if (tichLuy < x[midle])
            right = midle - 1;
        else
            left = midle + 1;
    } while (left <= right);
    return kq;
}
int TKNP_Giam(int x[MAX], int n, int tichLuy)
{
    int kq = -1, midle, left = 0, right = n-1;
    do
    {
        midle = (left + right) / 2;
        if (tichLuy == x[midle])
        {
            kq = midle;
            break;
        }
        else
        if (tichLuy < x[midle])
            left = midle + 1;
        else
            right = midle - 1;
    } while (left <= right);
    return kq;
}
void Xuat_TKNP_Theo_TichLuy(int tichLuy, sinhvien a[MAX], int n, int kq)
{
    if (kq == -1)
    {
        cout << "\nKhong co sinh vien trong danh sach co so TC tich luy == " << tichLuy << " :\n";
        return;
    }
    else
    {
        cout << "\nThong tin sinh vien trong danh sach co so TC tich luy == " << tichLuy << " :\n";
        Tieude();
        cout << endl;
        Xuat_SV(a[kq]);
        cout << endl;
    }
}
```


return;

}

}

E. Bài tập

Bài tập 1:

Giả sử có một danh sách nhân viên, mỗi nhân viên được lưu trữ các thông tin:

- Mã nhân viên, // Một chuỗi có 7 ký tự, không có ký tự trắng
- Họ nhân viên, // một chuỗi có không quá 10 ký tự
- Tên lót nhân viên, // một chuỗi có không quá 10 ký tự
- Tên nhân viên, // một chuỗi có không quá 10 ký tự
- Ngày tháng năm sinh của nhân viên, dạng : dd/mm/yyyy
- Địa chỉ, // một chuỗi có không quá 15 ký tự
- Lương, // Một số thực dương làm tròn, không quá 7 chữ số

Thực hiện các thao tác tìm kiếm trên danh sách nhân viên:

0. Tạo danh sách nhân viên
1. Xem danh sách nhân viên
2. Tìm kiếm theo họ, tên: Xuất tất cả các nhân viên trùng họ và tên cho trước.
3. Tìm kiếm năm sinh: Xuất tất cả các nhân viên cùng năm sinh.
4. Tìm kiếm theo họ, tên và năm sinh: Xuất tất cả các nhân viên trùng họ, tên cho trước và có năm sinh < x
5. Tìm kiếm theo tên và địa chỉ: Xuất tất cả nhân viên cùng tên và địa chỉ cho trước.
6. Tìm kiếm theo năm sinh và lương: Xuất các nhân viên có mức lương >= x và có năm sinh <= y
7. Tìm kiếm nhị phân theo mã nhân viên khi cho trước mã nhân viên (cần phải kiểm tra tính đơn điệu của dữ liệu)

Các chuỗi lưu trữ thông tin về họ, tên lót, tên, địa chỉ có thể gồm nhiều từ, các từ được nối với nhau bởi dấu gạch dưới. Về tên lót của nhân viên, nếu không có sẽ được thay thế bằng ký tự '_'.

Danh sách nhân viên được cho trong tập tin văn bản :

- test1.txt :

Ma NV	Ho	tLot	Ten	NTNSinh	Địa Chỉ	Lương
2523480	Vo	Minh	Duong	3 /1 /1980	Da_Lat	15000543
1134547	Hoang	Hoa	Tra	12/12/1987	Khanh_Hoa	18183210
2130007	Truong	Thanh	Hang	2 /2 /1980	Phu_Yen	15000543
1212345	Nguyen	Thi	Hang	10/8 /1987	Binh_Dinh	20232343
2223453	Le	Van	Trong	12/5 /1991	Hue	18183210
2003205	Van	Minh	Duong	10/10/1988	Da_Lat	18183210
1500348	Tran	Hoang	Hoa	15/10/1996	Binh_Dinh	18183210
2134007	Ly	Thi	Hang	18/7 /1988	Khanh_Hoa	20232343
2030007	Dinh	Thai	Duong	21/12/1980	Da_Lat	15000543
2011453	Le	Van	Tam	25/6 /1988	Hue	18183210
2022205	Trinh	Hoai	Duong	19/10/1991	Da_Lat	18183210
1652345	Nguyen	Minh	Tam	22/9 /1980	Binh_Dinh	20232343

- test2.txt :

Ma NV	Ho	tLot	Ten	NTNSinh	Địa Chỉ	Lương
LD12045	Nguyen	Tuan	Vo	1/1/1986	Lam_Dong	2500000
LD13210	Truong	Thi	Hoa	12/12/1985	Ninh_Thuan	3700000
LD13452	Tran	Ngoc	Ninh	2/2/1974	Khanh_Hoa	8000000
LD14432	Nguyen	_	Vo	10/8/1975	Phu_Yen	15000000
LD15332	Le	Thi	Lieu	12/5/1974	Binh_Dinh	12000000
LD22032	Van	Thi	Hoa	10/10/1984	Lam_Dong	6000000
LD22052	Vo	Ngoc	Hoa	1/10/1984	Lam_Dong	7000000
LD22140	Nguyen	Van	Vu	1/7/1990	Binh_Dinh	6200000
LD22145	Nguyen	Thi	Vo	10/6/1986	Khanh_Hoa	8000000
LD23045	Tran	Trong	Hieu	10/10/1991	Ha_Noi	15000000
LD24042	Ly	Van	Hoa	12/9/1983	Ninh_Thuan	30000000
LD30432	Le	_	Vo	12/12/1975	Lam_Dong	12000000

LAB 03. Thuật giải sắp xếp trong

A. Mục tiêu

Tìm hiểu và cài đặt các thuật giải sắp xếp trong

B. Yêu cầu

- Sinh viên tự ôn tập (Mục C)

- Buổi thực hành 3

+ 2 tiết đầu :

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Các bài 1 D(bỏ chức năng 8), 3D và 1E
 - Yêu cầu lưu trữ
 - ✓ Tạo thư mục **MaSV_HoVaTen_Lab03_Buoi3** chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Nén thư mục
 - Hình thức thu bài
 - ✓ Hệ thống thu bài trong phòng lab.

+ **Tiết cuối : Kiểm tra bài tập 1**

- Buổi thực hành 4 (3 t) :

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Các bài bài 1 D(chức năng 8), 2 D và 3E
 - Yêu cầu lưu trữ
 - ✓ Tạo thư mục **MaSV_HoVaTen_Lab03_Buoi4** chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Nén thư mục
 - Hình thức thu bài
 - ✓ Hệ thống thu bài trong phòng lab.

C. Ôn tập

- a. Phát biểu bài toán :
Cho dãy $a[0, \dots, n-1]$ gồm n số, sắp dãy này tăng dần.
- Input : a_0, a_2, \dots, a_{n-1}
int a[n];
- Output :
 - a tăng dần

Ghi chú : Tổng quát, sắp tăng một mảng cấu trúc theo một trường dữ liệu nào đó (có thứ tự).

- b. Các thuật giải sắp xếp trong:
 1. Chọn trực tiếp (Selection Straight sort)
 2. Chèn trực tiếp (Insertion Straight sort)
 3. Đổi chỗ trực tiếp (Interchange Straight sort)
 4. Nổi bọt (Bubble sort)
 5. Chèn nhị phân (Binary Insertion Sort)
 6. Quick sort
 7. Heap Sort
 8. Merge Sort
 9. Radix Sort

c. Ý tưởng các thuật giải :

1. Chọn trực tiếp (Selection Straight sort)

Thuật giải tiến hành trong $n-1$ bước, Với mọi $i = 0, \dots, n-2$, tìm min trong đoạn $[a_i, a_{n-1}]$ rồi đưa Min về đầu dãy $a_i, a_{i+1}, \dots, a_{n-1}$

2. Chèn trực tiếp (Insertion Straight sort):

Thuật giải tiến hành trong $n-1$ bước, Với mọi $i = 1, \dots, n-1$, tìm vị trí thích hợp của dãy con tăng dần $a_0, a_{i+1}, \dots, a_{i-1}$ để chèn $a[i]$ vào tạo ra dãy con $i+1$ phần tử a_0, a_{i+1}, \dots, a_i tăng dần.

Tìm vị trí thích hợp này bằng thuật giải tìm kiếm tuyến tính từ vị trí $i-1$ về 0.

3. Đổi chỗ trực tiếp (Interchange Straight sort):

Thuật giải tiến hành trong $n-1$ bước, Với mọi $i = 0, \dots, n-2$, tìm các phần tử sau a_i và tạo với a_i thành một cặp nghịch thế, rồi triệt tiêu các cặp nghịch thế này.

4. Nổi bọt (Bubble sort):

Thuật giải tiến hành trong $n-1$ bước, Với mọi bước $i = 0, \dots, n-2$, xuất phát từ cuối dãy đổi chỗ các cặp phần tử kế cận để đưa phần tử nhỏ nhất về đầu dãy.

5. Chèn nhị phân (Binary Insertion Sort):

Thuật giải tiến hành trong $n-1$ bước, Với mọi $i = 1, \dots, n-1$, tìm vị trí thích hợp của dãy con tăng dần $a_0, a_{i+1}, \dots, a_{i-1}$ để chèn $a[i]$ vào tạo ra dãy con $i+1$ phần tử a_0, a_{i+1}, \dots, a_i tăng dần.

Tìm vị trí thích hợp này bằng thuật giải tìm kiếm nhị phân

6. Thuật giải Quick sort:

Cải tiến thuật giải đổi chỗ trực tiếp bằng cách đổi chỗ các cặp không đúng thứ tự có thể xa nhau.

Thuật toán thực hiện việc lặp phân đoạn mảng đã cho thành 2 phần, nửa mảng bên trái $\leq x$, nửa mảng bên phải $\geq x$, cho đến khi các mảng con chỉ có 1 phần tử.

7. Thuật giải Heap Sort:

Cải tiến thuật giải chọn trực tiếp bằng cách tại mỗi bước, tận dụng thông tin các phép toán so sánh ở bước trước.

Từ heap ban đầu phát triển thành heap đầy đủ ban đầu, hoán vị 2 phần tử đầu và cuối để đưa giá trị lớn nhất của dãy về cuối dãy. Loại phần tử cuối (giá trị lớn nhất), và phần tử đầu a_0 .

Các phần tử còn lại là heap, bổ sung a_0 vào heap này để tạo thành một heap nhiều hơn 1 phần tử, rồi lặp lại các công việc trên để đưa giá trị lớn nhất về cuối.

Lặp quá trình trên đến khi chỉ còn 1 phần tử.

8. Thuật giải Merge Sort:

Thuật giải tiến hành nhiều bước lặp, mỗi bước gồm 2 giai đoạn:

- Tách luân phiên p phần tử của dãy đã cho vào 2 dãy trung gian.
- Trộn từng p phần tử của 2 dãy trung gian để tạo ra $2p$ phần tử tăng dần rồi lưu vào dãy ban đầu.
- p khởi tạo ban đầu bằng 1. Tại mỗi bước p khởi tạo lại bằng $2p$.

Thuật giải kết thúc khi $p > n$.

9. Thuật giải Radix sort:

Mô phỏng cách phân phối thư của bưu điện.

D. LUYỆN TẬP

Bài 1:

Viết chương trình tùy chọn sắp tăng dãy n số nguyên với các thuật giải:

0. Thoát khỏi chương trình
1. Tạo dữ liệu
2. Xem dữ liệu
3. Chọn trực tiếp (Selection Straight sort)
4. Chèn trực tiếp (Insertion Straight sort)
5. Đổi chỗ trực tiếp (Interchange Straight sort)
6. Nổi bọt (Bubble sort)
7. Chèn nhị phân (Binary Insertion Sort)
8. Radix

- Tập tin dữ liệu “*test1.txt*”:

8 //Kích thước mảng

12 2 8 5 1 6 4 15

- Tập tin dữ liệu “*test2.txt*”:

12 //Kích thước mảng
7013 8425 1239 428 1424 7009 4518 3252 9170 999 1725 701

Tổ chức chương trình theo thư viện hàm và có hệ thống tùy chọn menu.

Trong thư mục **MaSV_HoVaTen_Lab03_Buoi3**, tạo dự án Win32 Console Application mới tên là **Lab03_Bai1D**, có các tập tin thư viện **menu.h**, **thuvien.h**, tập tin chương trình **Program.cpp**

Thực hiện phát triển chương trình như lab 02, tham khảo tập tin thư viện **ThuVien.h** sau đây :

//Định nghĩa hằng : Khai báo kích thước mảng số nguyên

#define MAX 1000

//Định nghĩa kiểu dữ liệu

//Khai báo nguyên mẫu các hàm

//Định nghĩa các hàm

//Đọc dữ liệu trong tập tin filename chuyển qua mảng a

int File_Array(char *filename, int a[MAX], int &n)

```
{  
    ifstream in(filename);  
    if (!in)  
        return 0;  
    in >> n;  
  
    for (int i = 0; i < n; i++)  
        in >> a[i];  
    in.close();  
    return 1;  
}
```

//Hàm xuất dữ liệu

void Output(int a[MAX], int n)

```
{  
    int i;  
    for (i = 0; i < n; i++)  
        cout << a[i] << "t";  
}
```

//Hàm hoán vị

void HoanVi(int &a, int &b)

```
{  
    int tam = a;  
    a = b;  
    b = tam;  
}
```

//hàm chọn trực tiếp

//Tại mỗi bước đưa giá trị nhỏ nhất về đầu mảng

void Selection_L(int a[MAX], int n)

```
{  
    int i, j, cs_min;  
    for (i = 0; i < n - 1; i++)  
    {  
        cs_min = i;  
        for (j = i + 1; j < n; j++)  
            if (a[j] < a[cs_min])  
                cs_min = j;  
        HoanVi(a[i], a[cs_min]);  
    }  
}
```

//Copy a sang b

void Copy(int b[MAX], int a[MAX], int n)

```
{
    for (int i = 0; i < n; i++)
        b[i] = a[i];
}
//Tai moi buoc, chen PT hien hanh vao mang con ben trai tang dan
void Insertion_L(int a[MAX], int n)
{
    int i, x, pos;
    for (i = 1; i < n; i++)
    {
        x = a[i];
        for (pos = i - 1; (pos >= 0) && (a[pos] > x); pos--)
            a[pos + 1] = a[pos];
        a[pos + 1] = x;
    }
}
// Doi cho truc tiep :Tai moi buoc dua gia tri nho nhat ve dau mang
void Interchange_L(int a[MAX], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)
    {
        for (j = i + 1; j < n; j++)
            if (a[j] < a[i])
                HoanVi(a[i], a[j]);
    }
}
//Buble: Tai moi buoc dua GTNN ve dau mang
void Buble_L(int a[MAX], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)
    {
        for (j = n - 1; j > i; j--)
            if (a[j] < a[j - 1])
                HoanVi(a[j - 1], a[j]);
    }
}
//Chen nhi phan
void Binary_Insertion(int a[MAX], int n)
{
    int l, r, m;
    int i, j;
    int x;
    for (i = 1; i < n; i++)
    {
        x = a[i]; l = 0; r = i - 1;
        while (l <= r)
        {
            m = (l + r) / 2;
            if (x < a[m])
                r = m - 1;
            else
                l = m + 1;
        }
        for (j = i - 1; j >= l; j--)
            a[j + 1] = a[j];
        a[l] = x;
    }
}
```

```
}
```

//Sap theo co so

```
void Radix(int a[MAX], int n)
```

```
{
```

```
    int max, m;
```

```
    max = a[0]; m = 0;
```

```
    int k, i, j, du, thuong;
```

```
    int b0[MAX], b1[MAX], b2[MAX], b3[MAX], b4[MAX], b5[MAX],
```

```
        b6[MAX], b7[MAX], b8[MAX], b9[MAX];
```

```
    int p0, p1, p2, p3, p4, p5, p6, p7, p8, p9;
```

```
    //Tim max(a)
```

```
    for (i = 0; i < n; i++)
```

```
    if (a[i] > max)
```

```
        max = a[i];
```

```
    //Xã dinh so cac chu so cua max(a) : m
```

```
    while (max != 0)
```

```
    {
```

```
        max = max / 10;
```

```
        m++;
```

```
    }
```

```
    k = 0; //khoi tao chu so k = 0 : hang don vi
```

```
    while (k < m)
```

```
    {
```

```
        p0 = p1 = p2 = p3 = p4 = p5 = p6 = p7 = p8 = p9 = 0; //khoi tao chi so cua cac lo
```

```
        for (i = 0; i < n; i++)
```

```
        {
```

```
            //xac dinh chu so hang k cua a[i] : du
```

```
            thuong = a[i];
```

```
            for (j = 0; j <= k; j++)
```

```
            {
```

```
                du = thuong % 10;
```

```
                thuong = thuong / 10;
```

```
            }
```

```
            //Phan vao cac lo
```

```
            switch (du)
```

```
            {
```

```
                case 0: b0[p0++] = a[i];
```

```
                    break;
```

```
                case 1: b1[p1++] = a[i];
```

```
                    break;
```

```
                case 2: b2[p2++] = a[i];
```

```
                    break;
```

```
                case 3: b3[p3++] = a[i];
```

```
                    break;
```

```
                case 4: b4[p4++] = a[i];
```

```
                    break;
```

```
                case 5: b5[p5++] = a[i];
```

```
                    break;
```

```
                case 6: b6[p6++] = a[i];
```

```
                    break;
```

```
                case 7: b7[p7++] = a[i];
```

```
                    break;
```

```
                case 8: b8[p8++] = a[i];
```

```
                    break;
```

```
                case 9: b9[p9++] = a[i];
```

```
                    break;
```

```
            }
```

```
    }
```

```

} //Phan xong vào các k khi xét hàng k
//Nói lại theo trình tự để có dãy a tăng theo hàng k
j = 0;
for (i = 0; i < p0; i++)
    a[j++] = b0[i];
for (i = 0; i < p1; i++)
    a[j++] = b1[i];
for (i = 0; i < p2; i++)
    a[j++] = b2[i];
for (i = 0; i < p3; i++)
    a[j++] = b3[i];
for (i = 0; i < p4; i++)
    a[j++] = b4[i];
for (i = 0; i < p5; i++)
    a[j++] = b5[i];
for (i = 0; i < p6; i++)
    a[j++] = b6[i];
for (i = 0; i < p7; i++)
    a[j++] = b7[i];
for (i = 0; i < p8; i++)
    a[j++] = b8[i];
for (i = 0; i < p9; i++)
    a[j++] = b9[i];
k++; //xét tiếp hàng k lớn hơn, khi k = m thì dừng
}
}

```

Lưu ý là mỗi khi viết xong một hàm thì chạy chương trình kiểm tra lỗi

Bài 2:

Giả sử có một danh sách sinh viên, mỗi sinh viên được lưu trữ các thông tin:

- Mã sinh viên, //chuỗi có 7 ký tự, không có ký tự trắng
- Họ của sinh viên, //chuỗi có không quá 10 ký tự
- Tên lót, // chuỗi có không quá 10 ký tự
- Tên sinh viên, //chuỗi có không quá 10 ký tự
- Năm sinh, // số nguyên dương
- Lớp, // chuỗi có 5 ký tự, không có ký tự trắng
- Điểm trung bình, //số thực từ 0 đến 10
- Tích lũy, //số nguyên từ 0 đến 50

Các chuỗi lưu trữ thông tin về họ, tên lót, tên có thể gồm nhiều từ, các từ được nối với nhau bởi dấu gạch dưới.

- Tập tin dữ liệu “*test.txt*”:

Mã SV	Họ	Tên lót	Tên	Lớp	NSinh	Điểm TB	Tích Lũy
1412045	Nguyen	Tuan	Vo	CTK38	1996	6.2	30
1443210	Truong	Thi	Hoa	CTK38	1995	7.3	31
1423452	Tran	Ngoc	Ninh	CTK36	1994	8.6	32
1334432	Hoang	—	Hoa	CTK37	1995	5.2	36
1342332	Le	Thi	Lieu	CTK37	1994	4.6	37
1342032	Van	Thi	Hoa	CTK38	1994	6.6	38
1223052	Vo	Ngoc	Hoa	CTK36	1994	6.6	40
1312040	Nguyen	Van	Vu	CTK37	1996	6.2	41
1212045	Nguyen	Thi	Lieu	CTK37	1995	9.2	42
1313045	Tran	Trong	Hieu	CTK37	1993	4.1	43
1212042	Ly	Van	Hoa	CTK36	1993	5.3	44

Viết chương trình tùy chọn sắp tăng danh sách nhân viên theo trường “Điểm trung bình” với các thuật giải:

1. QuickSort
2. HeapSort
3. MergSort

Tạo thư mục **MaSV_HoVaTen_Lab03_Buoi4**.

Trong thư mục trên, tạo Project Lab04_Bai2D với 3 tập tin Thuvien.h, menu.h, program.cpp, hoàn thiện dần nội dung 3 tập tin này để có project hoàn chỉnh như lab 2.

- Tham khảo nội dung tập tin *Thuvien.h*:

```
//Sap danh sach nhan vien theo truong dtb
#define MAX 100
```

```
struct sinhvien
```

```
{
    char    maSV[8];
    char    hoSV[10];
    char    tenLot[10];
    char    ten[10];
    char    lop[6];
    int     namSinh;
    double  dtb;
    int     tichLuy;
};
```

```
//=====
int File_Array(char *filename, sinhvien a[MAX], int &n);
void Output_Arr(sinhvien a[MAX], int n);
void Output_Struct(sinhvien p);
void Heading();
//=====
void HoanVi(sinhvien &x, sinhvien &y);
void Copy(sinhvien b[MAX], sinhvien a[MAX], int n);
//=====
void QuickSort(sinhvien a[MAX], int n);
void Partition(sinhvien a[MAX], int l, int r);
void Shift(sinhvien a[MAX], int l, int r);
void Create_Heap(sinhvien a[MAX], int n);
void HeapSort(sinhvien a[MAX], int n);
void MergeSort(sinhvien F[MAX], int n);
void Merge(sinhvien F1[MAX], int h1, sinhvien F2[MAX], int h2, sinhvien F[MAX], int p);
void Distribution(sinhvien F[MAX], int n, sinhvien F1[MAX], int &h1, sinhvien F2[MAX], int &h2, int p);
//=====
//void nhap_mang(int a[MAX], int n);
void Output_Arr(int a[MAX], int n);
void File_Array(char *filename, int a[MAX], int &n);
void Ouput_Radix(int a[MAX], int n, int k);
//=====
```

```
//Cai dat cac ham chuc nang
```

```
//Thuat giai phan hoach
```

```
void Partition(sinhvien a[MAX], int l, int r)
{
    int i, j;
```



```
sinhvien x;
x = a[(l + r) / 2];
i = l; j = r;
do
{
    while (a[i].dtb < x.dtb)
        i++;
    while (a[j].dtb > x.dtb)
        j--;
    if (i <= j)
    {
        HoanVi(a[i], a[j]);
        i++; j--;
    }
} while (i <= j);

if (l < j)
    Partition(a, l, j);
if (i < r)
    Partition(a, i, r);
}

//Quick sort
void QuickSort(sinhvien a[MAX], int n)
{
    Partition(a, 0, n - 1);
}

//Heap sort
void Shift(sinhvien a[MAX], int l, int r)
{
    int i, j;
    sinhvien x;
    i = l; j = 2 * i + 1;
    x = a[i];
    while (j <= r)
    {
        if (j < r)
            if (a[j].dtb < a[j + 1].dtb)
                j = j + 1;
            if (a[j].dtb <= x.dtb)
                return;
        else
        {
            a[i] = a[j];
            i = j;
            j = 2 * i + 1;
            a[i] = x;
        }
    }
}

void Create_Heap(sinhvien a[MAX], int n)
{
    int l;
    l = (n - 1) / 2;
    while (l >= 0)
    {
        Shift(a, l, n - 1);
        l = l - 1;
    }
}
```

```
    }
}

void HeapSort(sinhvien a[MAX], int n)
{
    int r, i = 0;
    Create_Heap(a, n);
    r = n - 1;
    while (r > 0)
    {
        i++;
        HoanVi(a[0], a[r]);
        r = r - 1;
        Shift(a, 0, r);
    }
}

//Merge sort
void MergeSort(sinhvien F[MAX], int n)
{
    int p = 1, h1, h2;
    sinhvien F1[MAX], F2[MAX];
    int i = 1;
    while (p < n)
    {
        Distribution(F, n, F1, h1, F2, h2, p);
        Merge(F1, h1, F2, h2, F, p);

        i++;
        p = p * 2;
    }
}

//*****
void Distribution(sinhvien F[MAX], int n, sinhvien F1[MAX], int &h1, sinhvien F2[MAX], int &h2, int p)
{
    int i, k = 1, l = 0;
    h1 = 0; h2 = 0;
    do
    {
        i = 1;
        while (i <= p && l < n)
        {
            if (k == 1)
            {
                F1[h1++] = F[l];
            }
            else
            {
                F2[h2++] = F[l];
            }
            i++;
            l++;
        }
        k = 3 - k;
    } while (l < n);
}

//*****
void Merge(sinhvien F1[MAX], int h1, sinhvien F2[MAX], int h2, sinhvien F[MAX], int p)
{

```

```
int i1 = 0, i2 = 0, r1, r2;
int h = 0;
while (i1 < h1 && i2 < h2)
{
    r1 = r2 = 1;
    while ((r1 <= p) && (r2 <= p) && i1 < h1 && i2 < h2)
    {
        if (F1[i1].dtb <= F2[i2].dtb)
        {
            F[h++] = F1[i1];
            r1++;
            i1++;
        }
        else
        {
            F[h++] = F2[i2];
            r2++;
            i2++;
        }
    }
    while (i1 < h1 && r1 <= p)
    {
        F[h++] = F1[i1];
        i1++; r1++;
    }
    while (i2 < h2 && r2 <= p)
    {
        F[h++] = F2[i2];
        i2++; r2++;
    }
}
while (i1 < h1)
{
    F[h++] = F1[i1];
    i1++;
}
while (i2 < h2)
{
    F[h++] = F2[i2];
    i2++;
}
}

//=====
int File_Array(char *filename, sinhvien a[MAX], int &n)
{
    // tham khảo bài 2D lab02
}
//Xuat tieu de
void Heading()
{
    // tham khảo bài 2D lab02
```

```
}

//Xuat 1 sinh vien
void Output_Struct(sinhvien p)
{
    // tham khảo bài 2D lab02
}

//Xuat danh sach sinh vien
void Output_Arr(sinhvien a[MAX], int n)
{
    // tham khảo bài 2D lab02
}

void HoanVi(sinhvien &x, sinhvien &y)
{
    sinhvien t;
    t = x;
    x = y;
    y = t;
}

void Copy(sinhvien b[MAX], sinhvien a[MAX], int n)
{
    for (int i = 0; i < n; i++)
        b[i] = a[i];
}
```

Bài 3 :

Sắp tăng dần dãy số nguyên với các thuật toán sau :

1. Chọn trực tiếp (Selection Straight sort) : Selection_R(a,n)

Thuật giải tiến hành trong n-1 bước, Với mọi $i = 0, \dots, n-2$, tìm giá trị max trong đoạn $[a_0, a_{n-1-i}]$, rồi đưa Max về cuối dãy $a_0, a_{i+1}, \dots, a_{n-1-i}$

2. Thuật giải chọn 2 đầu : Selection_R_L(a,n)

Với mọi bước $i = 0 \rightarrow n/2 - 1$, trong đoạn $[a_i, a_{n-1-i}]$, đồng thời đưa giá trị nhỏ nhất về đầu dãy và đưa giá trị lớn nhất về cuối dãy.

3. Chèn trực tiếp (Insertion Straight sort): Insertion_R(a,n)

Thuật giải tiến hành trong n-1 bước, Với mọi $i = n-2 \downarrow 0$, tìm vị trí thích hợp của dãy con tăng dần $a_{i+1}, a_i, \dots, a_{n-1}$ để chèn $a[i]$ vào tạo ra dãy con $i+1$ phần tử $a_i, a_{i+1}, \dots, a_{n-1}$ tăng dần.

Tìm vị trí thích hợp này bằng thuật giải tìm kiếm tuyến tính từ vị trí $i+1$ đến $n-1$.

4. Đổi chỗ trực tiếp (Interchange Straight sort): Interchange_R(a,n)

Thuật giải tiến hành trong n-1 bước, Với mọi $j = n-1 \downarrow 1$, tìm các phần tử trước a_j và tạo với a_j thành một cặp nghịch thế, rồi triệt tiêu các cặp nghịch thế này.

5. Nổi bọt (Bubble sort): Buble_R(a,n)

Thuật giải tiến hành trong n-1 bước, Với mọi $j = n-1 \downarrow 1$, xuất phát từ đầu dãy đổi chỗ các cặp phần tử kế cận để đưa phần tử lớn nhất về đầu dãy.

6. Thuật giải Shaker sort (Cải tiến bubble)

Trong khi $l < r$, thực hiện trên đoạn $a[l, r]$ tuần tự 2 lượt :

- Lượt đi : từ biên phải tiến về trái, đổi chỗ 2 phần tử kế cận nếu cần thiết để đưa giá trị nhỏ nhất về đầu mảng, xác định lại biên trái l.
- Lượt về : từ biên trái tiến về phải, đổi chỗ 2 phần tử kế cận nếu cần thiết để đưa giá trị lớn nhất về đầu mảng, xác định

lại biên phải r.

- Sử dụng tập tin dữ liệu *test.txt* sau :

0	12	9	5	10	16	4	42	10	0	9	12
9	0	6	5	21	23	26	54	32	21	10	0
9	10	9	14	16	12	7	6	1	18	8	10

- Các hàm chức năng cài đặt như sau :

//1. Tại mỗi bước đưa giá trị lớn nhất về cuối mảng

```
void Selection_R(int a[MAX], int n)
{
    int i, j, cs_max;
    for (i = 0; i < n - 1; i++)
    {
        cs_max = n - 1 - i;
        for (j = n - 2 - i; j >= 0; j--)
            if (a[j] > a[cs_max])
                cs_max = j;
        HoanVi(a[n - 1 - i], a[cs_max]);
    }
}
```

//2. Tại mỗi bước đưa GTLN về cuối mảng, đưa GTNN về đầu mảng

```
void Selection_R_L(int a[MAX], int n)
{
    int i, j, cs_min, cs_max;
    for (i = 0; i < n / 2; i++)
    {
        cs_min = i;
        cs_max = n - 1 - i;
        for (j = i; j <= n - 1 - i; j++)
        {
            if (a[j] < a[cs_min])
                cs_min = j;
            if (a[j] > a[cs_max])
                cs_max = j;
        }
        if (cs_min == n - i - 1)
        {
            HoanVi(a[i], a[cs_min]);
            if (cs_max != i) //?
                HoanVi(a[cs_max], a[n - i - 1]);
        }
        else
        {
            HoanVi(a[cs_max], a[n - i - 1]);
            HoanVi(a[i], a[cs_min]);
        }
    }
}
```

//3. Insertion_R : Tại mỗi bước, chèn PT hiện hành vào mảng con bên phải tăng dần

```
void Insertion_R(int a[MAX], int n)
{
    int i, x, pos;
```

```
for (i = n - 2; i >= 0; i--)
{
    x = a[i];
    for (pos = i + 1; (pos < n) && (a[pos] < x); pos++)
        a[pos - 1] = a[pos];
    a[pos - 1] = x;
}
}

//4. Interchange_R: Tại mỗi bước đưa giá trị lớn nhất về cuối mảng
void Interchange_R(int a[MAX], int n)
{
    int i, j;
    for (j = n-1; j > 0; j--)
    {
        for (i = 0; i < j; i++)
            if (a[i] > a[j])
                HoanVi(a[i], a[j]);
    }
}

//5. Buble_R : Tại mỗi bước đưa GTLN về cuối mảng
void Buble_R(int a[MAX], int n)
{
    int i, j;
    for (j = n-1; j > 0; j--)
    {
        for (i = 0; i < n-1; i++)
            if (a[i] > a[i + 1])
                HoanVi(a[i + 1], a[i]);
        cout << "\nBuoc " << n - j << " : ";
        Output(a, n);
        cout << "\n";
    }
    cout << "\nCo " << n-1 << " bước thực hiện thuật giải.\n";
}

//6. Shaker :
void Shaker(int a[MAX], int n)
{
    int l = 0, r = n - 1;
    int k = n - 1;
    int j;
    while (l < r)
    {
        buoc++;
        j = r; //Khởi tạo j từ biên phải
        while (j > l)
        {
            if (a[j] < a[j - 1])
            {
                HoanVi(a[j], a[j - 1]);
                k = j;
                cout << "\nk = " << k;
            }
            j = j - 1;
        }
        l = k; //xác định lại biên trái l
        j = l; //khởi tạo j từ biên trái
        while (j < r)
        {
            if (a[j] > a[j + 1])
                HoanVi(a[j], a[j + 1]);
            k = j;
            cout << "\nk = " << k;
        }
        r = k;
    }
}
```

```

    {
        HoanVi(a[j], a[j + 1]);
        k = j;
    }
    j = j + 1;
}
r = k; //xác định lại biến phải l
}
}

```

E. Bài tập

(Bài 1 và Bài 2 dùng chung các tập tin dữ liệu test1.txt và test2.txt)

Bài 1:

Viết chương trình tùy chọn sắp tăng dãy n số nguyên gồm các chức năng:

0. Tạo dữ liệu
1. Xem dữ liệu
2. Thuật giải Chon Truc tiep - tai moi buoc dua GTNN ve dau mang
3. Thuật giải Chon Truc tiep - tai moi buoc dua GTLN ve cuoi mang
4. Thuật giải Chon hai dau
5. Thuật giải Chen Truc tiep - chen vao day con tang ben trai
6. Thuật giải Chen Truc tiep - chen vao day con tang ben phải
7. Thuật giải Chen nhì phan
8. Thuật giải Doi cho Truc tiep - tai moi buoc dua GTNN ve dau mang"
9. Thuật giải Doi cho Truc tiep - tai moi buoc dua GTLN ve cuoi mang"
10. Thuật giải Buble - tai moi buoc dua GTNN ve dau mang";
11. Thuật giải Buble - tai moi buoc dua GTLN ve cuoi mang";

Bài 2:

Viết chương trình tùy chọn sắp tăng dãy n số nguyên gồm các chức năng:

0. Tạo dữ liệu
1. Xem dữ liệu
2. Thuật giải Merge Sort
3. Thuật giải Heap Sort";
4. Thuật giải Radix Sort";
5. Thuật giải Quick Sort";

- Tập tin dữ liệu "test1.txt"

0 12 9 5 10 16 4 42

- Tập tin dữ liệu "test2.txt"

113 8425 1239 428 1424 7009 4518 3252 9070 999 1725 9701

Bài 3 và Bài 4 (dùng chung dữ liệu):

Giả sử có một danh sách nhân viên, mỗi nhân viên được lưu trữ các thông tin:

- Mã nhân viên, // chuỗi có 7 ký tự, không có ký tự trắng
- Họ của nhân viên, // chuỗi có không quá 10 ký tự
- Tên lót nhân viên, // một chuỗi có không quá 10 ký tự
- Tên nhân viên, // chuỗi có không quá 10 ký tự
- Địa chỉ, // chuỗi có không quá 15 ký tự
- Năm sinh, // số nguyên dương 4 ký số
- Lương, // số thực dương

- Tập tin dữ liệu "test.txt" lưu trữ danh sách nhân viên :

Mã NV	Họ	Tên lót	Tên	Địa Chỉ	NSinh	Lương
LD12045	Nguyen	Tuan	Vo	Lam_Dong	1980	25000000
LD13210	Ly	Van	Hoa	Ninh_Thuan	1985	30000000

Thực hành cấu trúc dữ liệu và thuật giải_2019-2020_HK2

LD13452	Tran	Ngoc	Ninh	Khanh_Hoa	1974	10000000
LD14432	Nguyen	–	Vo	Phu_Yen	1985	12000000
LD15332	Le	Thi	Lieu	Binh_Dinh	1974	10000000
LD22032	Van	Thi	Hoa	Lam_Dong	1984	10000000
LD22052	Vo_Hoang	Ngoc	Hoa	Lam_Dong	1984	70000000
LD22140	Tran	Vuong	Vo	Binh_Dinh	1990	12000000
LD22145	Le	Thi	Vo	Khanh_Hoa	1986	12000000
LD23045	Tran_Nguyen	Trong	Hieu	Ha_Noi	1991	25000000
LD24042	Ly	Van	Hoa	Ha_Noi	1983	30000000
LD30432	Nguyen	–	Vo	Lam_Dong	1975	12000000

Bài 3:

Viết chương trình tùy chọn sắp danh sách nhân viên theo yêu cầu :

- Dùng thuật giải chọn trực tiếp sắp danh sách tăng dần theo mã NV.
- Dùng thuật giải chèn trực tiếp sắp danh sách tăng dần theo địa chỉ.
- Dùng thuật giải Radix sắp danh sách tăng dần theo năm sinh

Bài 4:

Sắp danh sách nhân viên theo yêu cầu :

- Sắp tăng dần theo mức lương , dùng một trong các thuật giải :
 - Quick Sort,
 - Heap Sort,
 - Merge Sort
 - Shaker Sort
- Nếu cùng mức lương, sắp tăng dần theo tên
- Nếu cùng lương, tên, sắp tăng dần theo họ
- Nếu cùng lương, tên, họ, sắp tăng dần theo năm sinh..

Bài 5 :

Viết chương trình tùy chọn đếm số lượng các phép toán so sánh, các phép hoán vị trong các thuật giải :

1. Thuật giải Chọn Trục tiếp
2. Thuật giải Chèn Trục tiếp
3. Thuật giải Dôi cho Trục tiếp"
4. Thuật giải Buble

Với tập tin dữ liệu : “*test.txt*”

0	12	9	5	10	16	4	42	10	0	9	12
9	0	6	5	21	23	26	54	32	21	10	0
0	9	8	7	8	9	0	4	3	6	2	1
9	0	9	4	6	2	7	6	1	8	8	0
9	10	9	14	16	12	7	6	1	18	8	10
20	19	28	7	8	9	10	14	3	6	12	21
10	2	9	15	1	6	4	4	1	0	9	2

LAB 04. Danh sách liên kết đơn : Các thao tác cơ bản

A. Mục tiêu

- Tìm hiểu cách tổ chức kiểu DSLK đơn
- Thực hiện các thao tác cơ bản trên DSLK đơn.
- Thực hiện các ứng dụng liên quan.

B. Yêu cầu

- Sinh viên tự ôn mục C (ôn tập)

- Buổi thực hành 5 (3 t) :

- 2 tiết đầu : Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Các bài 1, 2 D (luyện tập); bài 2 E (Bài tập)
 - Yêu cầu lưu trữ
 - ✓ Tạo thư mục **MaSV_HoVaTen_Lab04** chứa trong ổ đĩa qui định.
 - ✓ Tạo project riêng cho mỗi bài, lưu trữ trong thư mục trên.
 - ✓ Mỗi project, xóa thư mục debug.
 - ✓ Nén thư mục
 - Hình thức thu bài
 - ✓ Nộp cho giáo viên qua mail : giaotiep2008@gmail.com.
- Tiết cuối : Sinh viên kiểm tra Bài tập 2

C. Ôn tập

- Cách tổ chức kiểu dữ liệu DSLK đơn*
- Cài đặt các thao tác thường dùng trên DSLK đơn*
- Tổ chức thư viện DSLK đơn*

I. Tổ chức kiểu dữ liệu DSLK đơn:

- DSLK đơn liên kết các phần tử hay còn gọi là nút (là các biến động) của DS theo một chiều đi từ phần tử đầu đến phần tử cuối, trong đó phần tử trước chứa thành phần liên kết (là con trỏ) chứa địa chỉ của phần tử kế sau. DSLK đơn được quản lý bằng 2 con trỏ: một chứa địa chỉ của phần tử đầu, một chứa địa chỉ phần tử cuối.
*Ta sẽ đặt tên Kiểu DSLK đơn (khi cài đặt) là : **LIST***
- Phần tử (biến động) của DS là một cấu trúc có 2 thành phần: Một thành phần chứa dữ liệu của phần tử, thành phần còn lại là con trỏ, được dùng để chứa địa chỉ của cấu trúc cùng kiểu (Cấu trúc tự trỏ)
*Ta sẽ đặt tên Kiểu phần tử (khi cài đặt) là : **NODE***
- Thành phần dữ liệu của phần tử (nút) của DS có kiểu dữ liệu là các kiểu đơn (ký tự, nguyên, thực) hay là các kiểu có cấu trúc tự đặt . . .
*Đặt tên Kiểu của thành phần dữ liệu của phần tử trong DS là : **data***
(data thay đổi tùy theo các ứng dụng)
data → NODE → LIST

II. Cài đặt kiểu dữ liệu DSLK đơn:

- Kiểu của thành phần dữ liệu trong DS:
(Giả sử là kiểu int)
typedef int data;
- Kiểu Phần tử (Nút) của DS:
 - Định nghĩa ban đầu

```
struct tagNode
{
    data          info;
    tagNode*      pNext;
};
```
 - Đổi lại tên:
typedef tagNode NODE;
- Kiểu DSLK đơn:

```
struct LIST
{
    NODE* pHead;//Con trỏ lưu địa chỉ phần tử đầu DSLK
    NODE* pTail; //Con trỏ lưu địa chỉ phần tử cuối DSLK
};
```

III. Các thao tác thường dùng trên DSLK đơn (trên các nút, trên danh sách)

(Giả sử ta đã cài đặt kiểu DSLK đơn như trên)

1. Tạo nút mới: **GetNode (x)** :
 - Chức năng: Tạo ra một phần tử (nút) của DSLK với thành phần dữ liệu là x (Thành phần liên kết của nút mới là con trỏ có giá trị NULL)
 - Input : x
 - Output : trả về con trỏ - lưu trữ địa chỉ của pt vừa tạo (nếu tạo thành công) - có giá trị NULL (nếu ngược lại).
 - Nguyên mẫu của hàm: `NODE* GetNode(Data x);`
2. Khởi tạo DS rỗng: **CreatList (l)**
 - Chức năng: Tạo ra một DSLK l rỗng.
 - Input : l
 - Output : l
 - Nguyên mẫu của hàm: `void CreatList(LIST &l);`
3. Kiểm tra DSLK l có rỗng: **IsEmpty(l)**
 - Chức năng: Kiểm tra DSLK l có rỗng rỗng?
 - Input : l
 - Output : 1; nếu l rỗng
0; ngược lại
 - Nguyên mẫu của hàm: `int IsEmpty(LIST l);`
4. Duyệt danh sách:
 - Chức năng: Duyệt danh sách từ đầu DS để xử lý dữ liệu của nút (có thể là xuất dữ liệu ra màn hình, đếm số nút, . . .)
 - Input : l
 - Nguyên mẫu của hàm: `void ProcessList (LIST l);`
5. Tìm nút có info là x:
 - Chức năng: Tìm trong DS có nút chứa thành phần Info là x?
 - Input : l, x
 - Output : p (con trỏ p chứa nút đầu tiên có info là x); nếu có
NULL; ngược lại
 - Nguyên mẫu của hàm: `NODE *Search(LIST l, Data x);`
6. Chèn nút (đã có) vào đầu DSLK đơn:
 - Chức năng: Chèn một nút (đã có trước) vào đầu DS (Chú ý: thành phần liên kết của nút này là con trỏ có giá trị NULL)
 - Input : l, new_ele
 - Output : l (Nút đầu là new_ele)
 - Nguyên mẫu của hàm: `void AddFirst(LIST &l, NODE* new_ele);`
7. Chèn một giá trị dữ liệu vào đầu DSLK đơn:
 - Chức năng: Tạo trước nút new_ele có info là x, con trỏ liên kết có giá trị NULL; sau đó Chèn nút này vào đầu DS.
 - Input : l, x (có kiểu data)
 - Output : - l (Nút đầu có info là x kiểu data)
- Con trỏ chứa địa chỉ của nút vừa tạo.
 - Nguyên mẫu của hàm: `NODE* InsertHead(LIST &l, data x);`
`void InsertHead(LIST &l, data x);`
8. Chèn nút (đã có) vào Cuối DSLK đơn:
 - Chức năng: Chèn một nút (đã có trước) vào cuối DS (Chú ý: thành phần liên kết của nút này là con trỏ có giá trị NULL)
 - Input : l, new_ele
 - Output : l (Nút cuối là new_ele)
 - Nguyên mẫu của hàm: `void AddTail(LIST &l, NODE *new_ele);`
9. Chèn một giá trị dữ liệu vào cuối DSLK đơn:
 - Chức năng: Tạo nút new_ele có info là x, con trỏ liên kết có giá trị NULL; sau đó chèn nút này vào cuối DS.
 - Input : l, x (có kiểu data)
 - Output : - l (Nút cuối có info là x kiểu data)
- Con trỏ chứa địa chỉ của nút vừa tạo.
 - Nguyên mẫu của hàm: `NODE* InsertTail (LIST &l, data x);`

void InsertTail (LIST &l, data x);

10. Chèn một nút (chưa có trước) vào sau nút do con trỏ q trỏ tới :
 - Chức năng: Tạo nút new_ele có info là x, con trỏ liên kết có giá trị NULL; sau đó chèn nút này vào sau nút do con trỏ q trỏ tới.
 - Input : l, q
 - Output : - 1 (Nút cuối có info là x kiểu data)
- Con trỏ chứa địa chỉ của nút vừa tạo.
 - Nguyên mẫu của hàm: void InsertAfter(LIST &l, NODE *q, data x);
11. Hủy nút đầu ra khỏi DSLK đơn:
 - Chức năng: Hủy nút đầu ra khỏi DSLK đơn
 - Input : l
 - Output : 1 (bớt nút đầu của l input)
 - Nguyên mẫu của hàm: void RemoveHead(LIST &l);
12. Hủy nút ở vị trí sau nút do con trỏ q trỏ tới :
 - Chức năng: Hủy nút sau nút có vị trí do con trỏ q trỏ tới.
 - Input : l, q
 - Output : 1 (bớt 1 nút)
 - Nguyên mẫu của hàm: void RemoveAfter (LIST &l, NODE *q);
13. Hủy nút có thành phần info là x :
 - Chức năng: Hủy nút có info là x.
 - Input : l, x
 - Output : 1; Nếu có nút
0; Nếu ngược lại
 - Nguyên mẫu của hàm: int RemoveNode(LIST &l, Data x);
14. Hủy toàn bộ danh sách
 - Chức năng: Hủy toàn bộ danh sách
 - Input : l
 - Output : 1 rỗng
 - Nguyên mẫu của hàm: void RemoveList(LIST &l);

D. LUYỆN TẬP

Bài 1:

Viết chương trình tùy chọn thực hiện các thao tác cơ bản trên danh sách liên kết đơn, dữ liệu của các nút trong danh sách là số nguyên:

0. Thoát khỏi chương trình
1. Tạo dữ liệu
2. Xem dữ liệu
- //Tìm kiếm
3. Tìm x - Có/Không
4. Tìm x - Tra về vị trí nút đầu tiên xuất hiện nếu có
5. Tìm x - Tra về vị trí nút cuối cùng xuất hiện nếu có
6. Tìm x - Xuất các vị trí xuất hiện nếu có";
- //Chen
7. Chen x vào đầu danh sách
8. Chen x vào cuối danh sách
9. Chen x vào danh sách sau nút có dữ liệu y xuất hiện đầu tiên
10. Chen x vào danh sách sau nút có dữ liệu y xuất hiện cuối cùng
- //Huy
11. Huy nút đầu danh sách
12. Huy nút cuối danh sách
13. Huy nút đầu tiên có x
14. Huy nút cuối cùng có x
15. Huy tất cả các nút có x
16. Huy toàn bộ danh sách";
- //Sắp xếp

17. Sắp tang-Chọn trực tiếp (hoán đổi dữ liệu);
18. Sắp tang- Chọn trực tiếp (hoán đổi liên kết);

Dữ liệu được cho trong các tập tin :

- “Test1.txt” :

9 7 1 7 -1 4 7 8 1 7 1 7

- “Test2.txt”

0 2 9 7 6 3 6 7 6 0

Thực hiện:

Trong thư mục **MaSV_HoVaTen_Lab04_D**, tạo dự án Win32 Console Application mới tên là **Lab04_Bai1D**, có các tập tin thư viện **menu.h**, **thuvien.h**, tập tin chương trình **Program.cpp**
Phát triển chương trình hoàn thiện từng bước theo các chức năng chương trình như lab 02.

- Tham khảo tập tin thư viện **ThuVien.h** sau đây :

```
//Định nghĩa hằng
#define MAX 100
//Định nghĩa kiểu dữ liệu
typedef int data; //Kiểu thành phần dữ liệu của nút
struct tagNode //Kiểu Nút
{
    data info;
    tagNode* pNext;
};
// Đổi lại tên kiểu Nút
typedef tagNode NODE;
//Kiểu DSLK đơn
struct LIST
{
    NODE* pHead; //Chứa dc nút đầu
    NODE* pTail; //Chứa dc nút cuối
};

//=====
//Khai báo nguyên mẫu các hàm
//Nhập, Xuất , hệ thống
void XuatDS(LIST l);
int File_List(char *f, LIST &l);
NODE* GetNode(data x);
void CreatList(LIST &l);
int IsEmpty(LIST l);
//=====
//Tìm kiếm
NODE *Search_First(LIST l, data x);
NODE *Search_End(LIST l, data x);
int Search_Node_x_First(LIST l, data x);
int Search_Node_x_End(LIST l, data x);
void Search_Node_x(LIST l, data x);
//=====
//Chèn
void AddFirst(LIST &l, NODE* new_ele);
void InsertHead(LIST &l, data x);
void AddTail(LIST &l, NODE *new_ele);
void InsertTail(LIST &l, data x);
void InsertAfter(LIST &l, NODE *q, data x);
void Insert_x_After_first_y(LIST &l, data y, data x);
```

```
void Insert_x_After_End_y(LIST &l, data y, data x);
//=====
//Huy
void RemoveHead(LIST &l);
void RemoveTail(LIST &l);
void RemoveList(LIST &l);
int RemoveNode_First(LIST &l, data x);
int RemoveNode_End(LIST &l, data x);
void Remove_x(LIST &l, data x);
//=====
//Sap xep
void List_Selection_Sort1(LIST &l);
void List_Selection_Sort2(LIST &l);

//=====
//Chuc nang khac
int SoNut_x(LIST l, data x);
int SoNut(LIST l);
void Copy(LIST &l1, LIST l);
void Hoanvi(data &x, data &y);

//=====
//Định nghĩa hàm
//=====
//Nhap, xuất, hệ thống
//=====
void XuatDS(LIST l)
{
    NODE *p;
    if (IsEmpty(l))
    {
        cout << "\nDS rong!\n";
        return;
    }
    p = l.pHead;
    while (p != NULL)
    {
        cout << p->info << "t";
        p = p->pNext;
    }
}

//Tap tin -> List : dung chen cuoi
int File_List(char *f, LIST &l)
{
    ifstream in(f); //Mo de doc
    if (!in)
        return 0;
    CreatList(l);
    data x;
    in >> x;
    InsertTail(l, x);
    while (!in.eof())
    {
        in >> x;
        InsertTail(l, x);
    }
    in.close();
    return 1;
}
```

```

}

//Ham tao nut co du lieu x, co next tro toi NULL
NODE* GetNode(data x)
{
    NODE *p;
    p = new NODE;
    if (p != NULL)
    {
        p->info = x;
        p->pNext = NULL;
    }
    return p;
}

//Tao DS rong
void CreatList(LIST &l)
{
    l.pHead = l.pTail = NULL;
}

//Kiem tra DS co rong
int IsEmpty(LIST l)
{
    if (l.pHead == NULL) // DS rỗng
        return 1;
    return 0;
}

//=====
//Tim kiem
//=====
// dang ket qua tim kiem: Co/Khong
// Co :Ham tra ve con tro chua dc cua nut dau tien co info la x. Khong co: tra ve NULL
NODE *Search_First(LIST l, data x)
{
    NODE *p;
    p = l.pHead;
    while ((p != NULL) && (p->info != x))
        p = p->pNext;
    return p;
}

//Tim nut co info la x:
//Khong co: ham tra ve -1
//Neu co, ham tra ve vi tri cua nut dau tien co info la x
//Nut dau tien dem tu 0
int Search_Node_x_First(LIST l, data x)
{
    NODE *p;
    int i = 0;
    p = l.pHead;
    while ((p != NULL) && (p->info != x))
    {
        p = p->pNext;
        i++;
    }
    if (p == NULL)
        i = -1;
}

```

```

        return i;
    }

//Ham tra ve con tro p chua dia chi nut cuoi cung co x (neu co).
NODE *Search_End(LIST l, data x)
{
    NODE *p,*q = NULL;
    p = l.pHead;
    while (p != NULL)
    {
        if (p->info == x)
            q = p;
        p = p->pNext;
    }
    return q;
}

//Neu co, ham tra ve vi tri cua nut cuoi cung co info la x
//Nut dau tien dem tu 0
int Search_Node_x_End(LIST l, data x)
{
    NODE *p;
    int Kq = -1, i = 0;
    p = l.pHead;
    while (p != NULL)
    {
        if (p->info == x)
            Kq = i;
        i++;
        p = p->pNext;
    }
    return Kq;
}

//Tim nut co info la x:
//Khong co: ham tra ve -1
//Neu co, ham xuat tat ca cac vi tri x xuat hien
//Nut dau tien dem tu 0
void Search_Node_x(LIST l, data x)
{
    NODE *p;
    int i = 0;
    p = Search_First(l,x);
    if (p == NULL)
    {
        cout << endl << x << " khong co trong danh sach";
        return;
    }
    cout << endl << x << " xuat hien trong danh sach tai cac vi tri :\n";

    p = l.pHead;
    while (p != NULL)
    {
        if (p->info == x)
            cout << i << "t";
        p = p->pNext;
        i++;
    }
}

//=====
//Chen

```

//Ham chen nut vao dau l

void AddFirst(LIST &l, NODE* new_ele)

```
{
    if (l.pHead == NULL)
    {
        l.pHead = new_ele;
        l.pTail = l.pHead;
    }
    else
    {
        new_ele->pNext = l.pHead;
        l.pHead = new_ele;
    }
}
```

//Ham chen x vao dau l

void InsertHead(LIST &l, data x)

```
{
    NODE* new_ele = GetNode(x);
    if (new_ele == NULL)
    {
        cout << "\nLoi cap phat bo nho! khong thuc hien duoc thao tac nay";
        return;
    }
    if (l.pHead == NULL)
    {
        l.pHead = new_ele; l.pTail = l.pHead;
    }
    else
    {
        new_ele->pNext = l.pHead;
        l.pHead = new_ele;
    }
}
```

//Chen cuoi

void AddTail(LIST &l, NODE *new_ele)

```
{
    if (IsEmpty(l))
    {
        l.pHead = new_ele; l.pTail = l.pHead;
    }
    else
    {
        l.pTail->pNext = new_ele;
        l.pTail = new_ele;
    }
}
```

//Chen cuoi

void InsertTail(LIST &l, data x)

```
{
    NODE* new_ele = GetNode(x);

    if (new_ele == NULL)
    {
        cout << "\nLoi cap phat bo nho! khong thuc hien duoc thao tac nay";
        return;
    }

    if (l.pHead == NULL)
```



```
{
    l.pHead = new_ele; l.pTail = l.pHead;
}
else
{
    l.pTail->pNext = new_ele;
    l.pTail = new_ele;
}
}
//Chen sau mot nut q
void InsertAfter(LIST &l, NODE *q, data x)
{
    NODE* new_ele = GetNode(x);
    if (new_ele == NULL)
    {
        cout << "\nLoi cap phat bo nho! khong thuc hien duoc thao tac nay";
        return;
    }
    if (q != NULL)
    {
        new_ele->pNext = q->pNext;
        q->pNext = new_ele;
        if (q == l.pTail)
            l.pTail = new_ele;
    }
    else
        AddFirst(l, new_ele);
}

// Chen x vao sau vi tri y xuất hiện đầu tiên trong danh sách nếu có, nếu không chen vao đầu danh sách
void Insert_x_After_first_y(LIST &l, data y, data x)
{
    NODE *q;
    q = Search_First(l, y); //tim y đầu tiên
    //chen sau q
    InsertAfter(l, q, x);
}

// Chen x vao sau vi tri y xuất hiện cuối cùng trong danh sách nếu có, nếu không chen vao đầu danh sách
void Insert_x_After_End_y(LIST &l, data y, data x)
{
    NODE *q;
    q = Search_End(l, y); //tim y đầu tiên
    //chen sau q
    InsertAfter(l, q, x);
}

//=====
//Huy

//Huy nút đầu
void RemoveHead(LIST &l)
{
    NODE *p;
    if (l.pHead == NULL)
    {
        cout << "\nDS rong!khong xoa duoc, chon thao tac khac!!!\n";
        return;
    }
}
```

```
p = l.pHead;
l.pHead = l.pHead->pNext;
delete p;
if (l.pHead == NULL)
    l.pTail = NULL;
}
//Hủy nút cuối
void RemoveTail(LIST &l)
{
    NODE *p, *q;
    if (l.pHead == NULL)
    {
        cout << "\nDS rong!khong xoa duoc, chon thao tac khac!!!\n";
        return;
    }
    //xác định địa chỉ nút đứng trước Tail : q
    p = l.pHead;
    q = NULL;
    while (p != l.pTail)
    {
        q = p;
        p = p->pNext;
    }
    l.pTail = q;
    delete p;
    if (l.pTail == NULL)
        l.pHead = NULL;
    else
        l.pTail->pNext = NULL;
}
//xóa nút đầu tiên có x
//kq = 0 : không có nút nào có x
//kq = 1, thành công
int RemoveNode_First(LIST &l, data x)
{
    NODE *p = l.pHead;
    NODE *q = NULL;

    while (p != NULL)
    {
        if (p->info == x)
            break;
        q = p; p = p->pNext;
    }
    if (p == NULL)
        return 0;
    //xóa nút sau q

    if (q != NULL)
    {
        if (p == l.pTail)
            l.pTail = q; //xóa nút cuối
        q->pNext = p->pNext;
    }
    else //xóa nút đầu
    {
        l.pHead = p->pNext;
        if (l.pHead == NULL)
            l.pTail = NULL;
    }
}
```

```

    }
    delete p;
    return 1;
}
//xoa nut cuoi cung co x
//kq = 0 : khong co nut nao co x
//kq = 1, thanh cong
int RemoveNode_End(LIST &l, data x)
{
    NODE *p = l.pHead;
    NODE *q = NULL;//truoc p
    NODE *r = NULL;
    NODE *t = NULL;//truoc r
    while (p != NULL)
    {
        if (p->info == x)
        {
            r = p;
            t = q;
        }
        q = p; p = p->pNext;
    }
    if (r == NULL)
        return 0;
    //xoa nut sau t
    if (t != NULL)
    {
        if (r == l.pTail)
            l.pTail = t; //xoa nut cuoi
        t->pNext = r->pNext;
    }
    else //xoa nut dau
    {
        l.pHead = r->pNext;
        if (l.pHead == NULL)
            l.pTail = NULL;
    }
    delete p;
    return 1;
}

//Huy cac nut co du lieu x
void Remove_x(LIST &l, data x)
{
    while (RemoveNode_First(l, x));
}

//Huy DS
void RemoveList(LIST &l)
{
    NODE *p;
    while (l.pHead != NULL)
    {
        p = l.pHead;
        l.pHead = p->pNext;
        delete p;
    }
    l.pTail = NULL;
}

```

```

}
    //sắp xếp
//Chọn trực tiếp: Hoán đổi dữ liệu
void List_Selection_Sort1(LIST &l)
{
    NODE *min;
    NODE *p, *q;
    p = l.pHead;
    while (p != l.pTail)
    {
        min = p;
        q = p->pNext;
        while (q != NULL)
        {
            if (q->info < min->info)
                min = q;
            q = q->pNext;
        }

        Hoanvi(min->info, p->info);
        p = p->pNext;
    }
}
//Sắp tang DS: Hoán đổi liên kết
void List_Selection_Sort2(LIST &l)
{
    LIST lRes; //DS kết quả trung gian
    NODE *min, //lưu nút đầu tiên có giá trị min tại mỗi bước
          *minprev; //Trước min
    NODE *p, //trước q
          *q; //duyet tìm min tại mỗi bước, mỗi bước xác định bởi pHead, sau mỗi bước l giảm một nút

    CreatList(lRes);
    while (l.pHead != NULL)
    {
        p = l.pHead;
        q = p->pNext; // sau p
        min = p;
        minprev = NULL;
        while (q != NULL)
        {
            if (q->info < min->info)
            {
                min = q;
                minprev = p;
            }
            p = q;
            q = q->pNext;
        }
        if (minprev != NULL)
            minprev->pNext = min->pNext;
        else //nút đầu là min, sẽ cắt nút này nên cập nhật pHead
            l.pHead = min->pNext;
        min->pNext = NULL;
        AddTail(lRes, min);
    }
    Copy(l, lRes);
}
//Đếm số lượng nút

```

```
int SoNut(LIST l)
{
    NODE *p;
    int i = 0;
    p = l.pHead;
    while (p != NULL)
    {
        p = p->pNext;
        i++;
    }
    return i;
}

//Copy l sang l1
void Copy(LIST &l1, LIST l)
{
    CreatList(l1);
    if (IsEmpty(l))
    {
        cout << "\nDS rong!";
        return;
    }
    NODE *p;
    data x;

    for (p = l.pHead; p != NULL; p = p->pNext)
    {
        x = p->info;
        InsertTail(l1, x);
    }
}

void Hoanvi(data &x, data &y)
{
    data t = x;
    x = y;
    y = t;
}

- Nội dung tập tin “menu.h” :
//soan thao he thong menu
void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, LIST &l);

void XuLyMenu(int menu, LIST &l)
{
    char filename[MAX];
    data x,y;
    NODE *p;
    int kq;
    LIST l1;
    switch (menu)
    {
    case 0:
        system("CLS");
        cout << "\n0. Thoat khoi chuong trinh\n";
        break;
    case 1:
        system("CLS");
        cout << "\n1. Tao du lieu";
```

```
do
{
    cout << "\nNhap ten tap tin, filename = ";
    _flushall();
    cin >> filename;
    kq = File_List(filename, l);
    if (!kq)
        cout << "\nLoi mo file ! nhap lai\n";
} while (!kq);

cout << "\nDanh sach nhap co " << SoNut(l) << " phan tu : \n\n";
XuatDS(l);
cout << endl;
break;

case 2:
    system("CLS");
    cout << "\n2. Xem du lieu";
    cout << "\nDanh sach l co " << SoNut(l) << " phan tu : \n";
    XuatDS(l);
    cout << endl;
    break;

case 3:
    system("CLS");
    cout << "\n3. Tim x Co/Khong?\n";
    cout << "\nDanh sach ban dau :\n";
    XuatDS(l);
    cout << "\nNhap x : ";
    cin >> x;
    p = Search_First(l, x);
    if (p != NULL)
        cout << endl << x << " co trong danh sach";
    else
        cout << endl << x << " khong co trong danh sach";
    cout << endl;
    break;

case 4:
    system("CLS");
    cout << "\n4. Tim x - Tra ve vi nut dau tien xuat hien neu co";
    cout << "\nDanh sach ban dau :\n";
    XuatDS(l);
    cout << "\nNhap x : ";
    cin >> x;
    kq = Search_Node_x_First(l, x);
    if (kq == -1)
        cout << endl << x << " khong co trong danh sach";
    else
        cout << endl << x << " xuat hien dau tien trong danh sach tai nut : " << kq;
    cout << endl;
    break;

case 5:
    system("CLS");
    cout << "\n5. Tim x - Tra ve vi nut cuoi cung xuat hien neu co";
    cout << "\nDanh sach ban dau :\n";
    XuatDS(l);
    cout << "\nNhap x : ";
    cin >> x;
    kq = Search_Node_x_End(l, x);
    if (kq == -1)
        cout << endl << x << " khong co trong danh sach";
```

else

```
    cout << endl << x << " xuất hiện cuối cùng trong danh sách tại nút : " << kq;  
    cout << endl;  
    break;
```

case 6:

```
    system("CLS");  
    cout << "\n6. Tìm x - Xuất các vị trí xuất hiện nếu có";  
    cout << "\nDanh sách ban đầu :\n";  
    XuatDS(l);  
    cout << "\nNhập x : ";  
    cin >> x;  
    Search_Node_x(l, x);  
    cout << endl;  
    break;
```

//Chen

case 7:

```
    system("CLS");  
    cout << "\n7. Chen x vào đầu danh sách";  
    cout << "\nDanh sách ban đầu :\n";  
    XuatDS(l);  
    cout << endl;  
    cout << "\nDanh sách có " << SoNut(l) << " phần tử.\n";  
    cout << "\nNhập x : ";  
    cin >> x;  
    InsertHead(l, x);  
    cout << "\nDanh sách đã chen " << x << " vào đầu danh sách:\n";  
    XuatDS(l);  
    cout << endl;  
    cout << "\nDanh sách mới có " << SoNut(l) << " phần tử.\n";  
    break;
```

case 8:

```
    system("CLS");  
    cout << "\n8. Chen x vào cuối danh sách";  
    cout << "\nDanh sách ban đầu :\n";  
    XuatDS(l);  
    cout << endl;  
    cout << "\nDanh sách có " << SoNut(l) << " phần tử.\n";  
    cout << "\nNhập x : ";  
    cin >> x;  
    InsertTail(l, x);  
    cout << "\nDanh sách đã chen " << x << " vào cuối danh sách:\n";  
    XuatDS(l);  
    cout << endl;  
    cout << "\nDanh sách mới có " << SoNut(l) << " phần tử.\n";  
    break;
```

case 9:

```
    system("CLS");  
    cout << "\n9. Chen x vào danh sách sau nút có dữ liệu y xuất hiện đầu tiên";  
    cout << "\nDanh sách ban đầu :\n";  
    XuatDS(l);  
    cout << endl;  
    cout << "\nDanh sách có " << SoNut(l) << " phần tử.\n";  
    cout << "\nNhập x cần chen : ";  
    cin >> x;  
    cout << "\nNhập y cần tìm kiếm : ";  
    cin >> y;  
    Insert_x_After_first_y(l, y, x);  
    cout << "\nDanh sách đã chen " << x << " vào danh sách sau vị trí đầu tiên " << y << " xuất hiện:\n";  
    XuatDS(l);
```

```
cout << endl;
cout << "\nDanh sach moi co " << SoNut(l) << " phan tu.\n";
break;
case 10:
    system("CLS");
    cout << "\n10. Chen x vao danh sach sau nut co du lieu y xuất hiện cuối cùng";
    cout << "\nDanh sach ban dau :\n";
    XuatDS(l);
    cout << endl;
    cout << "\nDanh sach co " << SoNut(l) << " phan tu.\n";
    cout << "\nNhap x can chen : ";
    cin >> x;
    cout << "\nNhap y can tìm kiếm : ";
    cin >> y;
    Insert_x_After_End_y(l, y, x);
    cout << "\nDanh sach đã chen " << x << " vào DS sau vị trí đầu tiên " << y << " xuất hiện:\n";
    XuatDS(l);
    cout << endl;
    cout << "\nDanh sach moi co " << SoNut(l) << " phan tu.\n";
    break;
//Huy
case 11:
    system("CLS");
    cout << "\n11. Huy nút đầu danh sach";
    cout << "\nDanh sach ban dau :\n";
    XuatDS(l);
    cout << endl;
    cout << "\nDanh sach co " << SoNut(l) << " phan tu.\n";
    RemoveHead(l);
    cout << "\nDanh sach sau khi huy nút đầu :\n";
    XuatDS(l);
    cout << endl;
    cout << "\nDanh sach moi co " << SoNut(l) << " phan tu.\n";
    break;
case 12:
    system("CLS");
    cout << "\n12. Huy nút cuối danh sach";
    cout << "\nDanh sach ban dau :\n";
    XuatDS(l);
    cout << endl;
    cout << "\nDanh sach co " << SoNut(l) << " phan tu.\n";
    RemoveTail(l);
    cout << "\nDanh sach sau khi huy nút cuối :\n";
    XuatDS(l);
    cout << endl;
    cout << "\nDanh sach moi co " << SoNut(l) << " phan tu.\n";
    break;
case 13:
    system("CLS");
    cout << "\n13. Huy nút đầu tiên có x";
    cout << "\nDanh sach ban dau :\n";
    XuatDS(l);
    cout << endl;
    cout << "\nDanh sach co " << SoNut(l) << " phan tu.\n";
    cout << "\nNhap x = ";
    cin >> x;
    kq = RemoveNode_First(l, x);
    if (!kq)
        cout << endl << x << " không có trong danh sach\n";
```



```
else
{
    cout << "\nDanh sach sau khi huy nut dau tien co " << x << ":\n";
    XuatDS(l);
    cout << endl;
    cout << "\nDanh sach moi co " << SoNut(l) << " phan tu.\n";
}
break;
case 14:
system("CLS");
cout << "\n14. Huy nut cuoi cung co x";
cout << "\nDanh sach ban dau :\n";
XuatDS(l);
cout << endl;
cout << "\nDanh sach co " << SoNut(l) << " phan tu.\n";
cout << "\nNhap x = ";
cin >> x;
kq = RemoveNode_End(l, x);
if (!kq)
    cout << endl << x << " khong co trong danh sach\n";
else
{
    cout << "\nDanh sach sau khi huy nut dau tien co " << x << ":\n";
    XuatDS(l);
    cout << endl;
    cout << "\nDanh sach moi co " << SoNut(l) << " phan tu.\n";
}
break;
case 15:
system("CLS");
cout << "\n15. Huy cac nut co x";
cout << "\nDanh sach ban dau :\n";
XuatDS(l);
cout << endl;
cout << "\nDanh sach co " << SoNut(l) << " phan tu.\n";
cout << "\nNhap x = ";
cin >> x;
if (!RemoveNode_First(l, x))
{
    cout << "\nKhong co " << x << " trong danh sach, danh sach khong doi\n";
}
else
{
    Remove_x(l, x);
    cout << "\nDanh sach sau khi huy cac nut co " << x << ":\n";
    XuatDS(l);
    cout << endl;
    cout << "\nDanh sach moi co " << SoNut(l) << " phan tu.\n";
}
break;
case 16:
system("CLS");
cout << "\n16. Huy toan bo danh sach";
Copy(l1, l);
cout << "\nDanh sach ban dau :\n";
XuatDS(l);
cout << endl;
cout << "\nDanh sach co " << SoNut(l) << " phan tu.\n";
RemoveList(l);
```

```
cout << endl<<"Danh sach sau khi huy tat ca cac phan tu:\n";
XuatDS(l);
cout << "\nPhuc hoi lai danh sach nhu truoc khi huy";
Copy(l, l1);
cout << "\nDanh sach da phuc hoi :\n";
XuatDS(l);
cout << endl;
cout << "\nDanh sach co " << SoNut(l) << " phan tu.\n";
break;
case 17:
system("CLS");
cout << "\n17. Sap tang-Chon truc tiep - Hoan doi du lieu";
Copy(l1, l);
cout << "\nDanh sach ban dau :\n";
XuatDS(l1);
cout << endl;
cout << "\nDanh sach co " << SoNut(l1) << " phan tu.\n";
List_Selection_Sort1(l1);
cout << endl << "Danh sach sau khi sap tang:\n";
XuatDS(l1);
cout << endl;
break;
case 18:
system("CLS");
cout << "\n18. Sap tang-Chon - Hoan doi lien ket";
Copy(l1, l);
cout << "\nDanh sach ban dau :\n";
XuatDS(l1);
cout << endl;
cout << "\nDanh sach co " << SoNut(l1) << " phan tu.\n";
system("PAUSE");
List_Selection_Sort2(l1);
cout << endl << "Danh sach sau khi sap tang:\n";
XuatDS(l1);
cout << endl;
break;
}
}
void XuatMenu()
{
//Tự viết
}
int ChonMenu(int soMenu)
{
//Tự viết
}
```

Bài 2:

Điểm quá trình môn Cấu trúc dữ liệu và Thuật giải 1 chiếm tối đa 4.5 điểm trong điểm tổng kết cuối cùng môn học (tương đương 45%), là tổng của 3 cột điểm thành phần sau :

- Điểm Bài tập I : chiếm tối đa 1 đ (tương đương 10 % điểm tổng kết)
- Điểm Bài tập II : chiếm tối đa 1.5 đ (tương đương 15% điểm tổng kết)
- Điểm kiểm tra giữa kỳ môn học : chiếm tối đa 2 đ (tương đương 20% điểm tổng kết)

Viết chương trình tính và xuất điểm quá trình của môn học cho các sinh viên đã đăng ký tham gia môn học.

Yêu cầu chương trình là xuất ra bảng điểm quá trình của sinh viên, thông tin của mỗi sinh viên bao gồm : Mã sinh viên, Họ, tên, năm sinh, điểm quá trình.

- Lý lịch sinh viên cho trong tập tin : “LLSV.test”

Mã SV	Họ	Tên	Năm Sinh	Nguyên quán
2410249	Nguyen	Canh	1986	Lam_Dong
2410271	Hoang	Dat	1985	Khanh_Hoa
2410272	Nguyen	Dat	1986	Ninh_Thuan
2410263	Khen	Di	1984	Lam_Dong
2410258	Kieu	Duyen	1986	Binh_Thuan
2413022	To	Duyen	1985	Lam_Dong
2413031	Do Van	Huong	1985	Binh_Dinh
2410295	Nguyen	Huy	1986	Quang_Ngai
2410302	Nguyen	Khanh	1984	Da_Nang
2413041	Vo	Kiet	1983	Binh_Thuan
2410316	Lam	Le	1986	Lam_Dong
2410312	K	Luyt	1986	Lam_Dong
2413047	Nguyen	Mai	1985	Ninh_Thuan
2410327	Dang	Nga	1986	Binh_Thuan
2410336	Nguyen	Ngan	1985	Binh_Thuan
2410349	Vo	Phong	1986	Binh_Dinh
2410387	Vo	Tam	1984	Quang_Ngai
2410401	Nguyen	Thao	1985	Binh_Thuan
2410415	Mong	Vu	1986	Lam_Dong
2410420	Luu	Y	1986	Da_Nang

- Các cột điểm thành phần của điểm quá trình cho trong tập tin "DiemQT.txt" :

Mã SV	Điểm BT I	Điểm BT II	Điểm KTGK
2410249	1	1.5	1.5
2410271	0.5	1.5	2
2410272	1	1	1
2410263	1	1.5	2
2410258	0.5	1.5	1
2413022	1	1	1.5
2413031	1	1.5	2
2410295	0.5	1.5	1.5
2410302	1	1.5	2
2413041	1	1.5	1.5
2410316	01	1.5	1.5
2410312	0.5	1.5	2
2413047	01	1	2
2410327	0.5	1.5	2
2410336	0	0.5	1.5
2410349	1	1.5	2
2410387	1	1.5	2
2410401	0.5	0.5	1.5
2410415	1	1.5	1
2410420	0.5	1.5	2

Thực hiện :

Bước 1:

Tạo dự án Win32 Console Application mới. Đặt tên là Lab04_Bai2D (chứa trong thư mục **MaSV_HoVaTen_Lab04_D**), với các tập tin rỗng **thuvien.h** (trong **Header Files**), **Program.cpp** (trong **Source File**). Không tạo tập tin menu.h, vì không có tùy chọn chức năng.

Bước 2 :

- Trong **Program.cpp** soạn thảo nội dung :

#include <iostream>

#include <fstream>

```
#include <string.h>
#include <iomanip>
#include <conio.h>
using namespace std;

#include "Thuvien.h"
void ChayChuongTrinh();
```

```
int main()
{
    ChayChuongTrinh();
    return 1;
}
```

```
void ChayChuongTrinh()
{
}
```

Nhấn Ctrl+F5 chạy chương trình để kiểm tra lỗi

Bước 3.

- Trong tập tin thuvien.h, soạn thảo các nội dung : định nghĩa kiểu dữ liệu, các thao tác hệ thống, nhập xuất dữ liệu, các hàm chức năng thực hiện theo yêu cầu bài toán.

Lưu ý là ta cần chạy thường xuyên chương trình để kiểm tra lỗi.

- Nội dung tập tin *thuvien.h* :

```
//Định nghĩa kiểu dữ liệu
//Kiểu dữ liệu list: LL
struct LLSinhVien //Kiểu sinh viên
{
    char maSV[8];
    char ho[15];
    char ten[10];
    int namSinh;
    char nguyenQuan[15];
};

struct tagNodeLL
{
    LLSinhVien info;
    tagNodeLL *pNext;
};

typedef tagNodeLL NODELL;
struct LL
{
    NODELL *pHead;
    NODELL *pTail;
};

//=====
//Kiểu dữ liệu list: BD (Bảng điểm)
struct DiemTP
{
    char maSV[8];
    double dBT1;
    double dBT2;
    double dQT;
    double dQT;
};
```

```
struct tagNodeDiem
{
    DiemTP info;
    tagNodeDiem *pNext;
};

typedef tagNodeDiem NODEBD;

struct BD
{
    NODEBD *pHead;
    NODEBD *pTail;
};

//=====
//Khai bao nguyên mẫu
NODELL* GetNode_LL(LLSinhVien x);
void CreatList_LL(LL &l);
void InsertTail_LL(LL &l, LLSinhVien x);
void TapTin_List_LL(char *filename, LL &l);
//=====
void TapTin_List_BD(char *filename, BD &l);
void InsertTail_BD(BD &l, DiemTP x);
void CreatList_BD(BD &l);
NODEBD* GetNode_BD(DiemTP x);
//=====
void XuatBangDiem(LL l, BD g);
void TieuDe();
//=====
//Định nghĩa hàm
//Tạo nút mới LL
NODELL* GetNode_LL(LLSinhVien x)
{
    NODELL *p;
    p = new NODELL;
    if (p != NULL)
    {
        p->info = x;
        p->pNext = NULL;
    }
    return p;
}

//Khởi tạo DSLK đơn lý lịch rỗng
void CreatList_LL(LL &l)
{
    l.pHead = l.pTail = NULL;
}

//Chèn x vào cuối ds
void InsertTail_LL(LL &l, LLSinhVien x)
{
    NODELL* new_ele = GetNode_LL(x);
    if (new_ele == NULL)
    {
        cout << "\nKhông đủ bộ nhớ";
        _getch();
        return;
    }
}
```

```
if (l.pHead == NULL)
{
    l.pHead = new_ele; l.pTail = l.pHead;
}
else
{
    l.pTail->pNext = new_ele;
    l.pTail = new_ele;
}
}

//Chuyen du lieu tap tin ly lich sinh vien sang List l
void TapTin_List_LL(char *filename, LL &l)
{
    LLSinhVien x;
    ifstream in(filename);
    if (!in)
    {
        cout << "\nLoi mo file !\n";
        _getch();
        return;
    }
    CreatList_LL(l);
    char maSV[8];
    char ho[15];
    char ten[10];
    int namSinh;
    char nguyenQuan[15];

    in >> maSV; strcpy_s(x.maSV, 8, maSV);
    in >> ho; strcpy_s(x.ho, 15, ho);
    in >> ten; strcpy_s(x.ten, 10, ten);
    in >> namSinh; x.namSinh = namSinh;
    in >> nguyenQuan; strcpy_s(x.nguyenQuan, 15, nguyenQuan);
    InsertTail_LL(l, x);

    while (!in.eof())
    {
        in >> maSV; strcpy_s(x.maSV, 8, maSV);
        in >> ho; strcpy_s(x.ho, 15, ho);
        in >> ten; strcpy_s(x.ten, 10, ten);
        in >> namSinh; x.namSinh = namSinh;
        in >> nguyenQuan; strcpy_s(x.nguyenQuan, 15, nguyenQuan);
        InsertTail_LL(l, x);
    }
    in.close();
}

//=====
//Bang diem
//Tao nut moi BD
NODEBD* GetNode_BD(DiemTP x)
{
    NODEBD *p;
    p = new NODEBD;
    if (p != NULL)
    {
        p->info = x;
    }
}
```

```
        p->pNext = NULL;
    }
    return p;
}

//Khởi tạo DSLK đơn bang điểm rong
void CreatList_BD(BD &l)
{
    l.pHead = l.pTail = NULL;
}

//Chèn x vào cuối ds
void InsertTail_BD(BD &l, DiemTP x)
{
    NODEBD* new_ele = GetNode_BD(x);
    if (new_ele == NULL)
    {
        cout << "\nKhông đủ bộ nhớ";
        _getch();
        return;
    }

    if (l.pHead == NULL)
    {
        l.pHead = new_ele; l.pTail = l.pHead;
    }
    else
    {
        l.pTail->pNext = new_ele;
        l.pTail = new_ele;
    }
}

//Chuyển dữ liệu tập tin điểm thành phần sang List BD
//Chuyển dữ liệu tập tin điểm thành phần sang List BD
void TapTin_List_BD(char *filename, BD &l)
{
    DiemTP x;
    ifstream in(filename);
    if (!in)
    {
        cout << "\nLỗi mở file !\n";
        _getch();
        return;
    }
    CreatList_BD(l);
    char maSV[8];
    double dBT1;
    double dBT2;
    double dGK;
    double dQT;

    in >> maSV; strcpy_s(x.maSV, maSV);
    in >> dBT1; x.dBT1 = dBT1;
    in >> dBT2; x.dBT2 = dBT2;
    in >> dGK; x.dGK = dGK;
    dQT = dBT1 + dBT2 + dGK; x.dQT = dQT;

    InsertTail_BD(l, x);
}
```

```

while (!lin.eof())
{
    in >> maSV; strcpy_s(x.maSV, maSV);
    in >> dBT1; x.dBT1 = dBT1;
    in >> dBT2; x.dBT2 = dBT2;
    in >> dGK; x.dGK = dGK;
    dQT = dBT1 + dBT2 + dGK; x.dQT = dQT;

    InsertTail_BD(l, x);
}
in.close();
}

```

// Xuất kết quả bảng điểm

//Xuất tiêu đề

void TieuDe()

```

{
    int i;
    cout << endl;
    cout << ':';
    for (i = 1; i <= 54; i++)
        cout << '=';
    cout << ':'<<endl;

    cout << setiosflags(ios::left);
    cout << ':';
    cout << setw(10) << "Ma SV"
        << ':'
        << setw(16) << "Ho"
        << setw(11) << "Ten"
        << ':'
        << setw(6) << "NSinh"
        << ':'
        << setw(8) << "DiemGK"
        << ':';

    cout << endl;
    cout << ':';
    for (i = 1; i <= 54; i++)
        cout << '=';
    cout << ':';
}

```

void XuatBangDiem(LL l, BD g)

```

{
    TieuDe();
    NODELL *p = l.pHead;
    NODEBD *q=g.pHead;
    while (p != NULL && q != NULL)
    {
        cout << endl << ':';
        cout << setiosflags(ios::left)
            << setw(10) << p->info.maSV
            << ':'
            << setw(16) << p->info.ho
            << setw(11) << p->info.ten
            << ':'

```



```

        << setw(6) << p->info.namSinh
        << ':';
        << setw(8) << setprecision(2) << setiosflags(ios::fixed) << q->info.dQT
        << ':';
        p = p->pNext;
        q = q->pNext;
    }

    cout << endl;
    cout << ':';
    for (int i = 1; i <= 54; i++)
        cout << '=';
    cout << ':' << endl;
}

```

Bước 4.

Tạo các tập tin dữ liệu : LLSV.txt, diemQT.txt

Bước 5.

Cập nhật lại hàm ChayChuongTrinh() trong Program.cpp như sau :

```

void ChayChuongTrinh()
{
    LL l;
    BD g;
    TapTin_List_LL("LLSV.txt", l);
    TapTin_List_BD("DiemQT.txt", g);
    XuatBangDiem(l,g);
    _getch();
}

```

Nhấn Ctrl+F5 chạy chương trình kiểm tra lỗi, kiểm tra kết quả.

Kết thúc chương trình

Bài 3 :

Giả sử có một danh sách nhân viên, mỗi nhân viên được lưu trữ các thông tin:

- Mã nhân viên, // chuỗi có 7 ký tự, không có ký tự trắng
- Họ của nhân viên, // chuỗi có không quá 10 ký tự
- Tên lót nhân viên, // một chuỗi có không quá 10 ký tự
- Tên nhân viên, // chuỗi có không quá 10 ký tự
- Địa chỉ, // chuỗi có không quá 15 ký tự
- Năm sinh, // số nguyên dương 4 ký số
- Lương, // số thực dương

- Tập tin dữ liệu "test.txt" lưu trữ danh sách nhân viên :

Mã NV	Họ	Tên lót	Tên	Địa Chỉ	NSinh	Lương
LD12045	Nguyen	Tuan	Vo	Lam_Dong	1980	25000000
LD13210	Ly	Van	Hoa	Ninh_Thuan	1985	30000000
LD13452	Tran	Ngoc	Ninh	Khanh_Hoa	1974	10000000
LD14432	Nguyen	-	Vo	Phu_Yen	1985	12000000
LD15332	Le	Thi	Lieu	Binh_Dinh	1974	10000000
LD22032	Van	Thi	Hoa	Lam_Dong	1984	10000000
LD22052	Vo	Ngoc	Hoa	Lam_Dong	1984	70000000
LD22140	Tran	Vuong	Vo	Binh_Dinh	1990	12000000
LD22145	Le	Thi	Vo	Khanh_Hoa	1986	12000000
LD23045	Tran	Trong	Hieu	Ha_Noi	1991	25000000
LD24042	Ly	Van	Hoa	Ha_Noi	1983	30000000
LD30432	Nguyen	-	Vo	Lam_Dong	1975	12000000

Viết chương trình tùy chọn thực hiện các thao sau trên danh sách nhân viên:

1. Tách danh sách nhân viên thành 2 danh sách, danh sách đầu gồm những nhân viên có lương $\leq x$, danh sách sau gồm những nhân viên còn lại.
2. Tách danh sách nhân viên theo cách luân phiên từng nhân viên theo thứ tự vào 2 danh sách.
3. Đảo ngược danh sách.
4. Sắp danh sách tăng dần theo tên, tên trùng thì tăng dần theo họ; tên và họ trùng thì tăng dần theo tên lót.

Thực hiện :

Tạo dự án Win32 Console Application Lab05_Bai3D (chứa trong thư mục đã tạo), với các tập tin rỗng **thuvien.h**, **menu.h** (trong **Header Files**), **Program.cpp** (trong **Source File**).

Tạo tập tin dữ liệu **“test.txt”** theo yêu cầu chương trình.

Bổ sung dần vào các tập tin cho đến khi hoàn chỉnh, mỗi lần bổ sung chức năng mới cần chạy ngay chương trình để kiểm tra.

- Tham khảo Nội dung tập tin **“thuvien.h”**:

//Định nghĩa kiểu dữ liệu : LIST

struct nhanvien

{

 char maNV[8];

 char hoNV[10];

 char tenLot[10];

 char ten[10];

 char diachi[12];

 int namSinh;

 double lương;

};

typedef nhanvien data;

struct tagNode

{

 data info;

 tagNode *pNext;

};

typedef tagNode NODE;

struct LIST

{

 NODE *pHead;

 NODE *pTail;

};

//=====

NODE* GetNode(data x);

void CreatList(LIST &l);

int IsEmpty(LIST l);

void InsertHead(LIST &l, data x);

void InsertTail(LIST &l, data x);

//=====

void TapTin_List(char *filename, LIST &l);

void Xuat_DS NV(LIST l);

void Xuat_NV(data p);

void TieuDe();

//=====

void Tach_Luong_x(LIST l, double x);

void Tach_LuanPhien(LIST l);

void DaoNguoc_DS(LIST l);

void Remove_Ten(LIST &l, char ten[10]);

int RemoveNode_First(LIST &l, char ten[10]);

void List_SapTang_Ten_Ho_TLot(LIST &l);

void List_Selection_Sort(LIST &l);

void Hoanvi(data &x, data &y);

```
//=====
//Tao nut moi
NODE* GetNode(data x)
{
    NODE *p;
    p = new NODE;
    if (p != NULL)
    {
        p->info = x;
        p->pNext = NULL;
    }
    return p;
}
//Khoi tao DSLK don rong
void CreatList(LIST &l)
{
    l.pHead = l.pTail = NULL;
}
//Kiem tra danh sach co rong
int IsEmpty(LIST l)
{
    if (l.pHead == NULL)
        return 1;
    return 0;
}
//Chen x vao dau ds
void InsertHead(LIST &l, data x)
{
    NODE* new_ele = GetNode(x);
    if (new_ele == NULL)
    {
        cout << "\nKhong du bo nho";
        system("PAUSE");
        return;
    }
    if (l.pHead == NULL) //DS rong
    {
        l.pHead = new_ele;
        l.pTail = l.pHead;
    }
    else
    {
        new_ele->pNext = l.pHead; //chen vao dau DS
        l.pHead = new_ele;
    }
}
//Chen x vao cuoi ds
void InsertTail(LIST &l, data x)
{
    NODE* new_ele = GetNode(x);
    if (new_ele == NULL)
    {
        cout << "\nKhong du bo nho";
        system("CLS");
        return;
    }
    if (l.pHead == NULL)
    {
        l.pHead = new_ele; l.pTail = l.pHead;
    }
}
```

```
    }
    else
    {
        l.pTail->pNext = new_ele;
        l.pTail = new_ele;
    }
}

//=====
//Chuyen du lieu tap tin dang list
void TapTin_List(char *filename, LIST &l)
{
    data x;
    ifstream in(filename);
    if (!in)
    {
        cout << "\nLoi mo file !\n";
        system("PAUSE");
        return;
    }
    CreatList(l);
    char maNV[8];
    char hoNV[10];
    char tenLot[10];
    char ten[10];
    char diachi[12];
    int namSinh;
    double luong;
    in >> maNV; strcpy_s(x.maNV, maNV);
    in >> hoNV; strcpy_s(x.hoNV, hoNV);
    in >> tenLot; strcpy_s(x.tenLot, tenLot);
    in >> ten; strcpy_s(x.ten, ten);
    in >> diachi; strcpy_s(x.diachi, diachi);
    in >> namSinh; x.namSinh = namSinh;
    in >> luong; x.luong = luong;
    InsertTail(l, x);
    while (!in.eof())
    {
        in >> maNV; strcpy_s(x.maNV, maNV);
        in >> hoNV; strcpy_s(x.hoNV, hoNV);
        in >> tenLot; strcpy_s(x.tenLot, tenLot);
        in >> ten; strcpy_s(x.ten, ten);
        in >> diachi; strcpy_s(x.diachi, diachi);
        in >> namSinh; x.namSinh = namSinh;
        in >> luong; x.luong = luong;
        InsertTail(l, x);
    }
    in.close();
}

//Xuat tieu de
void TieuDe()
{
    //Tu viet
}

//Xuat 1 nhan vien
void Xuat_NV(data p)
{
    //Tu viet
}
```

```
//Xuat danh sach nhan vien
void Xuat_DSNV(LIST l)
{
    //Tu viet
}

void Tach_Luong_x(LIST l, double x)
{
    NODE *p;
    LIST l1, l2;

    p = l.pHead;
    if (p == NULL)
    {
        cout << "\nDS l rong";
        system("PAUSE");
        return;
    }

    CreatList(l1);
    CreatList(l2);
    while (p != NULL)
    {
        if (p->info.luong <= x)
            InsertTail(l1, p->info);
        else
            InsertTail(l2, p->info);
        p = p->pNext;
    }
    cout << "\n- Danh sach l1 (luong <= " << x << "):\n";
    Xuat_DSNV(l1);
    cout << "\n- Danh sach l2 (luong > " << x << "):\n";
    Xuat_DSNV(l2);
    cout << endl;
}

void Tach_LuanPhien(LIST l)
{
    NODE *p;
    LIST l1, l2;

    p = l.pHead;
    if (p == NULL)
    {
        cout << "\nDS l rong";
        system("PAUSE");
        return;
    }
    int k = 1; //khoa
    CreatList(l1);
    CreatList(l2);
    while (p != NULL)
    {
        if (k == 1)
            InsertTail(l1, p->info);
        else
            InsertTail(l2, p->info);
        p = p->pNext;
        k = 3 - k;
    }
}
```

```
cout << "\n- Danh sach l1:\n";
Xuat_DSNV(l1);
cout << "\n- Danh sach l2 :\n";
Xuat_DSNV(l2);
cout << endl;
}
void DaoNguoc_DS(LIST l)
{
    NODE *p;
    LIST l1;

    p = l.pHead;
    if (p == NULL)
    {
        cout << "\nDS l rong";
        system("PAUSE");
        return;
    }
    CreatList(l1);
    while (p != NULL)
    {
        InsertHead(l1, p->info);
        p = p->pNext;
    }
    cout << "\n- Danh sach dao nguoc cua l:\n";
    Xuat_DSNV(l1);
}
//Sap tang theo ten
void List_Selection_Sort(LIST &l)
{
    NODE *min;
    NODE *p, *q;
    p = l.pHead;
    while (p != l.pTail)
    {
        min = p;
        q = p->pNext;
        while (q != NULL)
        {
            if (_stricmp(q->info.ten, min->info.ten) < 0)
                min = q;
            q = q->pNext;
        }
        Hoanvi(min->info, p->info);
        p = p->pNext;
    }
}
void List_SapTang_Ten_Ho_TLot(LIST &l)
{
    List_Selection_Sort(l);
    NODE *p, *q;
    for (p = l.pHead; p != l.pTail; p = p->pNext)
        for (q = p->pNext; q != NULL; q = q->pNext)
            if (_stricmp(p->info.ten, q->info.ten) == 0)
                if (_stricmp(p->info.hoNV, q->info.hoNV) > 0)
                    Hoanvi(q->info, p->info);

    for (p = l.pHead; p != l.pTail; p = p->pNext)
```

```
for (q = p->pNext; q != NULL; q = q->pNext)
    if (_strcmpi(p->info.ten, q->info.ten) == 0 && _strcmpi(p->info.hoNV, q->info.hoNV) == 0)
        if (_strcmpi(p->info.tenLot, q->info.tenLot) > 0)
            Hoanvi(q->info, p->info);
}
```

- Nội dung các tập tin **menu.h**, **program.cpp** : Tự viết

E. Bài tập

Bài 1:

Viết chương trình tùy chọn thực hiện các chức năng sau đây trên danh sách liên kết đơn, dữ liệu của các nút trong danh sách là số nguyên:

0. Thoat khỏi chương trình
1. Tạo danh sách
2. Xem danh sách
3. Tính giá trị nhỏ nhất của danh sách
4. Tính giá trị lớn nhất của danh sách
5. Đếm giá trị x xuất hiện bao nhiêu lần trong danh sách
6. Tính tổng các giá trị trong danh sách
7. Tính tổng các giá trị khác nhau trong danh sách
8. Xuất các giá trị khác nhau trong danh sách và số lần xuất hiện tương ứng
9. Sắp danh sách tang dần bằng thuật giải chèn trực tiếp
10. Sắp danh sách theo yêu cầu:
 - Giá trị 0 xuất hiện tại các nút đầu danh sách
 - Tiếp theo là các nút có giá trị âm giảm dần
 - Cuối cùng là các nút có giá trị dương tăng dần

Dữ liệu cho trong các tập tin :

- "Test1.txt" :

9 7 -1 7 -1 0 7 -8 1 7 1 0

- "Test2.txt" :

0 2 -9 7 6 -3 6 7 6 0

Bài 2: (TÍNH LƯƠNG)

Giả sử quy tắc tổ chức quản lý nhân viên của một công ty như sau :
Thông tin về một nhân viên bao gồm lý lịch và bảng chấm công :

- Lý lịch nhân viên :

- Mã nhân viên : chuỗi 7 ký tự
- Họ và Tên nhân viên : chuỗi không quá 15 ký tự
- Địa chỉ : Chuỗi không quá 15 ký tự
- Trình độ văn hoá : 1 ký số
(1 = cấp 1 ; 2 = cấp 2 ; 3 = cấp 3 ; 4 = đại học; 5 = cao học, 6 = Tiến sĩ)
- Lương căn bản : số nguyên $\leq 6\,500\,000$

- Chấm công nhân viên :

- Số ngày nghỉ có phép trong tháng : số ≤ 28
- Số ngày nghỉ không phép trong tháng : số ≤ 28
- Số ngày làm thêm trong tháng : số ≤ 8

- Quy tắc tính lương :

Lương thực lĩnh = Lương căn bản + Phụ trội

Trong đó phụ trội tính như sau :

- Trình độ văn hoá = cao học : Phụ trội = +10% Lương căn bản
- Trình độ văn hoá = tiến sĩ : Phụ trội = +20% Lương căn bản

Thực hành cấu trúc dữ liệu và thuật giải_2019-2020_HK2

- Làm thêm: Phụ trội = +4% Lương căn bản/ngày
- Nghỉ không phép : Phụ trội = -4%Lương căn bản/ngày
- Nghỉ có phép (Nếu số ngày nghỉ ≥ 15) : Phụ trội = -10%Lương căn bản

Viết chương trình tính lương nhân viên trong tháng, yêu cầu bảng lương của nhân viên bao gồm các thông tin sau : Mã nhân viên, Họ và Tên nhân viên, Lương căn bản, phụ trội, lương thực lĩnh; và danh sách tăng dần theo lương thực lĩnh, nếu lương thực lĩnh bằng nhau thì tăng dần theo mã nhân viên.

- Lý lịch nhân viên cho trong tập tin **"lly.txt"**:

Mã NV	Họ và Tên	Địa chỉ	TĐVH	Lương CB
LD30432	Nguyen_Vo	Lam_Dong	4	4000000
LD12045	Nguyen_Tuan	Lam_Dong	1	2500000
LD13210	Ly_Van_Hoa	Ninh_Thuan	4	4000000
LD13452	Tran_Ngoc	Khanh_Hoa	6	5500000
LD14432	Nguyen_Vo	Phu_Yen	4	4000000
LD15332	Le_Thi_Lieu	Binh_Dinh	4	4000000
LD22032	Van_Thi_Hoa	Lam_Dong	2	3000000
LD22052	Vo_Hoang	Lam_Dong	1	3000000
LD22140	Tran_Vuong	Binh_Dinh	3	3200000
LD22145	Le_Thi_Vo	Khanh_Hoa	3	3200000
LD23045	Tran_Nguyen	Ha_Noi	6	6000000
LD24042	Ly_Van_Hoa	Ha_Noi	4	4000000

- Bảng chấm công nhân viên cho trong tập tin **"chamcong.txt"**:

Mã NV	Số ngày nghỉ có phép	Số ngày nghỉ 0 phép	Số ngày làm thêm
LD30432	4	2	2
LD12045	3	0	4
LD13210	4	2	2
LD13452	15	3	5
LD14432	0	2	2
LD15332	2	2	2
LD22032	3	1	2
LD22052	20	2	0
LD22140	2	2	2
LD22145	1	3	1
LD23045	1	4	1
LD24042	0	2	2

Bài 3 :

Với dữ liệu như trong phần D, bài 3, Viết chương trình tùy chọn sắp tăng danh sách nhân viên theo năm sinh :

1. Thuật giải Chọn trực tiếp
2. Thuật giải Chèn trực tiếp
3. Thuật giải Buble
4. Thuật giải merge sort
5. Thuật giải Radix

Bài 4:

(Là bài 5 lab 1, nhưng tổ chức cấu trúc dữ liệu bằng danh sách liên kết đơn và dữ liệu chuẩn bị trong các tập tin)

Giải bóng đá ngoại hạng Anh 2016 - Premier League 2016 – không có nhà tài trợ, do công ty Football Association Premier League quản lý và điều hành, có 20 câu lạc bộ bóng đá chuyên nghiệp vương quốc Anh tham dự, đá vòng tròn 2 lượt đi về xếp hạng với thể thức giải đấu như sau :

- **Tính điểm cho một trận đấu** : đội thắng – 3 điểm; hòa – 1 điểm ; thua – 0 điểm

- **Thực hiện xếp hạng dựa vào kết quả thi đấu của các đội, các tiêu chí xếp loại chung cuộc (ưu tiên theo thứ tự) sau :**
 - **Tổng điểm**
 - **Nếu có nhiều đội bằng điểm, xét tiếp các tiêu chí theo thứ tự ưu tiên sau :**
 - **Hiệu số bàn thắng thua**
 - **Số bàn thắng**
- **Đội có kết quả cao hơn được xếp trên, các đội có các kết quả trùng nhau sẽ được xếp cùng hạng.** Nếu việc xếp cùng hạng đó ảnh hưởng đến tới chức vô địch, xuống hạng hay giành quyền tham dự một giải đấu khác, một trận play-off sẽ được diễn ra trên sân trung lập để xác định thứ hạng.
- **Kết quả xếp hạng chung cuộc :**
 - **Chọn được 3 đội có kết quả cao nhất để trao giải : Vô địch (HC vàng), Nhì (HC bạc), ba (HC đồng)**
 - **3 đội có điểm thấp nhất sẽ phải xuống hạng (chơi ở giải kế dưới vào năm sau).**
- **Thẻ thực thi đấu tham khảo thêm tại :**

https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_b%C3%B3ng_%C4%91%C3%A1_Ngo%E1%BA%A1i_h%E1%BA%A1ng_Anh#Th.C3.A0nh_1.E1.BA.ADp

- **Thông tin về các đội tham dự, lịch thi đấu và kết quả các trận đấu xem trên trang web :**
 - <http://bongdanet.vn/bang-xep-hang-bong-da/ngoai-hang-anh/2016-2017>
 - <http://www.livescore.com>

Viết chương trình tùy chọn thực hiện các chức năng sau :

0. Thoát khỏi chương trình
1. Xem lịch thi đấu vòng j
2. Xem lịch thi đấu toàn giải
3. Xem kết quả các trận đấu vòng j
4. Xem kết quả các trận đấu đến vòng j (từ vòng 1 đến vòng j)
5. Xem kết quả xếp hạng đến vòng j
6. Xem kết quả xếp hạng lượt đi
7. Xem kết quả chung cuộc

Một số gợi ý về tổ chức chương trình :

Chương trình tổ chức theo thư viện và có hệ thống tùy chọn menu (xem lab 1). Thực hiện theo cách khi thêm một phần mới vào chương trình (định nghĩa dữ liệu, hàm,...) ta cần chạy kiểm tra cú pháp ngay để dễ sửa lỗi. Để chạy được, phải có tập tin program.cpp chứa hàm main.

Chương trình tổ chức với 1 tập tin chương trình .cpp, 5 tập tin thư viện .h, và 2 tập tin văn bản dữ liệu

Ta tổ chức các tập tin thư viện sau :

1. Tập tin **01_Header_KQTD.h** :
 - Định nghĩa dữ liệu: Định nghĩa kiểu kết quả trận đấu, danh sách liên kết đơn kết quả trận đấu,
 - Các thao tác cơ bản trên danh kết quả trận đấu : tạo nút mới, tạo DSLK đơn các trận đấu rỗng, chèn dữ liệu vào cuối danh sách, chuyển dữ liệu tập tin kết quả các trận đấu vào danh sách kết quả các trận đấu.
2. Tập tin **02_ThuVien_KQTD.h**:
 - Xuất lịch thi đấu mỗi vòng, toàn giải
 - Xuất kết quả các trận đấu tại một vòng, hay từ đầu đến vòng nào đó
 - Xác định vòng thi đấu mới nhất
3. Tập tin **03_Header_KQDB.h**:
 - Định nghĩa dữ liệu: Định nghĩa kiểu kết quả xếp hạng đội bóng, danh sách liên kết đơn kết quả đội bóng,
 - Các thao tác cơ bản trên danh kết quả các đội bóng : tạo nút mới, tạo DSLK đơn các trận đấu rỗng, chèn dữ liệu vào cuối danh sách, chuyển dữ liệu tập tin khởi tạo kết quả các đội bóng vào danh sách kết quả các trận đấu.
 - Tạo kết quả đội bóng cho các vòng đấu từ kết quả các trận đấu.
4. Tập tin **04_XepHang.h** :
 - Đánh số xếp hạng các đội theo điều lệ giải
5. Tập tin **05_Menu.h** :
 - Tổ chức hệ thống tùy chọn thực hiện chức năng
6. Tập tin **Program.cpp** :
 - Điều khiển thực hiện chương trình
7. Tập tin dữ liệu **kqtd.txt** :

- Lưu trữ kết quả các trận đấu (theo mỗi vòng)
- 8. Tập tin dữ liệu ***Khoitao_KQDB.txt*** :
 - Lưu trữ khởi tạo kết quả các đội bóng.

Lưu ý rằng : với các tập tin ***01_Header_KQTD.h, 02_ThuVien_KQTD.h, 05_Menu.h, Program.cpp, kqtd.txt*** thì thực hiện được 5 chức năng đầu của bài toán.

Sau đây là nội dung một số tập tin :

1. Tập tin ***01_Header_KQTD.h*** :

```
//Định nghĩa kiểu kết quả trận đấu - data1
//Định nghĩa LIST KQTD
//Định nghĩa các thao tác cơ bản : tạo nút mới, tạo list rỗng, chen
//Chuyển dữ liệu tập tin vào list KQTD
//Định nghĩa kiểu KetQuaTranDau, là kiểu của thành phần dữ liệu của nút
struct KetQuaTranDau
{
    int    vong;           //Vòng đấu
    char   ngay[9];        //Ngày thực hiện trận đấu dưới dạng chuỗi : dd-mm-yy
    char   gio[6];         //Giờ thực hiện trận đấu, dưới dạng chuỗi xx:xx
    char   cNha[15];       //Tên Đội chủ nhà
    int    btChu;          //Số bàn thắng đội chủ nhà ghi được
    int    btKhach;        //Số bàn thắng đội khách ghi được (bàn thắng trên sân đối phương của trận)
    char   khach[15];      //Tên đội khách
};
typedef KetQuaTranDau data1;
struct tagNODE1
{
    data1 kqtd;
    tagNODE1 *pNext;
};
typedef tagNODE1 NODE1;
//Kiểu DSLK đơn kết quả các trận đấu
struct LIST_KQTD
{
    NODE1 *pHead;
    NODE1 *pTail;
};
//=====
//Các thao tác cơ bản trên LIST_KQTD
//Tạo nút mới
NODE1* GetNODE1(data1 x)
{
    NODE1 *p;
    p = new NODE1;
    if (p != NULL)
    {
        p->kqtd = x;
        p->pNext = NULL;
    }
    return p;
}
//Khởi tạo DSLK đơn rỗng
void CreatList_KQTD(LIST_KQTD &l)
{
    l.pHead = l.pTail = NULL;
}
//Chen x vào cuối ds
void InsertTail1(LIST_KQTD &l, data1 x)
{
    NODE1* new_ele = GetNODE1(x);
    if (new_ele == NULL)
```

```

{
    cout << "\nKhong du bo nho";
    _getch();
    return;
}

if (l.pHead == NULL)
{
    l.pHead = new_ele; l.pTail = l.pHead;
}
else
{
    l.pTail->pNext = new_ele;
    l.pTail = new_ele;
}
}

//=====
//Chuyen du lieu tap tin chua ket qua cac tran dau vao list
void TapTin_List_KQTD(char *filename, LIST_KQTD &l)
{
    ifstream in(filename);
    if (!in)
    {
        cout << "\nLoi mo file !\n";
        _getch();
        return;
    }
    CreatList_KQTD(l);

    int    vong;           //Vong dau
    char   ngay[9];        //Ngay thuc hien tran dau duoi dang chuoi : dd-mm-yy
    char   gio[6];         //Gio thuc hien tran dau, duoi dang chuoi xx:xx
    char   cNha[15];       //Ten Doi chu nha
    int    bTChu;          //So ban thang doi chu nha ghi duoc
    int    bTKhach;        //So ban thang doi khach ghi duoc (ban thang tren san doi phuong cua tran)
    char   khach[15];      //Ten doi khach
    data1 x;
    in >> vong; x.vong = vong;
    in >> ngay; strcpy_s(x.ngay, 9, ngay);
    in >> gio; strcpy_s(x.gio, 6, gio);
    in >> cNha; strcpy_s(x.cNha, 15, cNha);
    in >> bTChu; x.bTChu = bTChu;
    in >> bTKhach; x.bTKhach = bTKhach;
    in >> khach; strcpy_s(x.khach, 15, khach);
    InsertTail1(l, x);
    while (!in.eof())
    {
        in >> vong; x.vong = vong;
        in >> ngay; strcpy_s(x.ngay, 9, ngay);
        in >> gio; strcpy_s(x.gio, 6, gio);
        in >> cNha; strcpy_s(x.cNha, 15, cNha);
        in >> bTChu; x.bTChu = bTChu;
        in >> bTKhach; x.bTKhach = bTKhach;
        in >> khach; strcpy_s(x.khach, 15, khach);
        InsertTail1(l, x);
    }
    in.close();
}

```

2. Tập tin **02_ThuVien_KQTD.h**:

```
//Lich thi dau - Ket qua tran dau
//xuat lich thi dau, ket qua cac tran dau
//Dinh nghia hang
#define MAX 100//kich thước khai mang cau truc
#define NGANGDOI '='
#define NGANGDON '-'
//Dinh nghia cac ham
```

```
////////////////////////////////////
//          Xuat lich thi dau          //
////////////////////////////////////
```

```
void XuatNganDoi()
```

```
{
    int i;
    cout << '\n';
    for (i = 1; i < 59; i++)
        cout << NGANGDOI;
    cout << '\n';
}
```

```
//Xuat tieu de lich thi dau cac vong
```

```
void XuatTieuDe_LTD()
```

```
{
    XuatNganDoi();
    cout << endl;
    cout << setiosflags(ios::left)
        << ": "
        << setw(4) << "Vong"
        << ": "
        << setw(9) << "Ngay"
        << ": "
        << setw(6) << "Gio"
        << ": "
        << setw(15) << "Chu Nha"
        << ": "
        << setw(15) << "Khach"
        << ":\n";
    cout << endl;
    XuatNganDoi();
}
```

```
//Xuat lich thi dau 1 tran
```

```
void XuatLTD_1TD(data1 p)
```

```
{
    cout << setiosflags(ios::left)
        << ": "
        << setw(4) << p.vong
        << ": "
        << setw(9) << p.ngay
        << ": "
        << setw(6) << p.gio
        << ": "
        << setw(15) << p.cNha
        << ": "
        << setw(15) << p.khach
        << ":\n";
}
```

```
//Xuat lich thi dau cua vong j
```

```
void Xuat_LTD_Vong_j(LIST_KQTD l, int vong)
```

```
{
    int k, dem = 0;
    cout << "\n          Lich thi dau vong " << vong;
```

```

cout << endl;
XuatTieuDe_LTD();
NODE1 *p;
for (p = l.pHead; p != NULL; p = p->pNext)
{
    if (p->kqtd.vong == vong)
    {
        cout << endl;
        XuatLTD_1TD(p->kqtd);
        dem++;
        if (dem == 10)
            break;
        if (p->pNext == NULL)
            break;
        if (_stricmp(p->kqtd.ngay, p->pNext->kqtd.ngay) != 0)
        {
            cout << endl;
            cout << ':';
            for (k = 1; k < 59; k++)
                cout << NGANGDON;
            cout << ':';
        }
    }
}
cout << endl;
XuatNganDoi();
}
//Xuat lich thi dau den vong j
void Xuat_LTD_DenVong_j(LIST_KQTD l, int vong)
{
    int j;
    cout << endl;
    system("CLS");
    for (j = 1; j <= vong; j++)
    {
        Xuat_LTD_Vong_j(l, j);
        cout << endl;
        _getch();
    }
}

////////////////////////////////////
//          Xuat ket qua cac tran dau          //
////////////////////////////////////

//Xuat tieu de Ket qua tran dau
void XuatTieuDe_KQTD()
{
    int i;
    cout << ':';
    for (i = 1; i < 70; i++)
        cout << NGANGDOI;
    cout << ':';
    cout << endl;
    cout << setw(4) << "Vong"
    cout << ':'
    cout << setw(9) << "Ngay"
    cout << ':'
    cout << setw(6) << "Gio"

```

```

    << ':';
    << setw(15) << "Chu Nha"
    << ':';
    << setw(15) << "    Ty So    "
    << ':';
    << setw(15) << "Khach"
    << ':';
    cout << endl;
    cout << ':';
    for (i = 1; i < 70; i++)
        cout << NGANGDOI;
    cout << ':';
}
//Xuat Ket qua 1 tran dau
void XuatKQ_1TD(data1 p)
{
    cout << setiosflags(ios::left)
    << ':';
    << setw(4) << p.vong
    << ':';
    << setw(9) << p.ngay
    << ':';
    << setw(6) << p.gio
    << ':';
    << setw(15) << p.cNha
    << ':';
    << setw(7) << p.bTChu
    << ':';
    << setw(7) << p.bTKhach
    << ':';
    << setw(15) << p.khach
    << ':';
}
//Xuat ket qua cac tran dau cua vong j
void Xuat_KQTD_Vong_j(LIST_KQTD l, int vong)
{
    NODE1 *p;
    int k;
    int dem = 0;
    cout << "\n          Ket qua cac tran dau vong " << vong;
    cout << endl;
    XuatTieuDe_KQTD();

    for (p = l.pHead; p != NULL; p = p->pNext)
    {
        if (p->kqtd.vong == vong)
        {
            cout << endl;
            XuatKQ_1TD(p->kqtd);
            dem++;
            if (dem == 10)
                break;
            if (p->pNext == NULL)
                break;
            if (_stricmp(p->kqtd.ngay, p->pNext->kqtd.ngay) != 0)
            {
                cout << endl;
                cout << ':';
                for (k = 1; k < 70; k++)

```

```

        cout << NGANGDON;
        cout << ':';
    }
}
}
cout << endl;
cout << ':';
for (k = 1; k < 70; k++)
    cout << NGANGDOI;
cout << ':';
}

//Xuat ket qua cac tran dau den vong j
void Xuat_KQTD_DenVong_j(LIST_KQTD l, int vong)
{
    int j;
    cout << "\nKet qua cac tran dau den vong " << vong << " :";
    cout << endl;
    for (j = 1; j <= vong; j++)
    {
        Xuat_KQTD_Vong_j(l,j);
        cout << endl;
        _getch();
    }
}

//Xac dinh vong thi dau moi nhat
int VongTD_MoiNhat(LIST_KQTD l)
{
    int vong = 38;
    NODE1 *p;
    for (p = l.pHead; p != NULL; p = p->pNext)
        if (p->kqtd.bTChu == -1)
        {
            vong = p->kqtd.vong - 1;
            break;
        }
    return vong;
}

```

3. Tập tin **03_Header_KQDB.h**:

```

//Định nghĩa data2 : kiểu KQXepHang
//Định nghĩa LIST_KQXH và các thao tác cơ bản
//Tính Kết quả các đội bóng nhưng chưa xếp hạng,
//Định nghĩa kiểu dữ liệu mới
struct KetQuaXepHang
{
    char    tenDoi[15];    //ten doi
    int     st;            //so tran da dau (-> vong moi nhat)
    int     sTThang;       //So tran thang
    int     sTHoa;         //So tran hoa
    int     sTThua;        //So tran thua
    int     sBThang;       //So ban thang
    int     sBThua;        //So ban thua
    int     hieuSo;        //Hieu so ban thang thua
    int     diem;          //Diem
    int     xh;            //Xep hang
}

```

```
//Khoi Tao Bang Ket qua cac doi bong :
//chuyen tap tin khoi tao ket qua doi bong vao List Ket qua doi bong
```



```
//=====
void KhoiTao_List_KQXH(LIST_KQXH &h)
{
    ifstream in("Khoitao_KQDB.txt");
    if (!in)
    {
        cout << "\nLoi mo file !\n";
        _getch();
        return;
    }
    CreatList_KQXH(h);
    char tenDoi[15]; //ten doi
    int st; //so tran da dau (-> vong moi nhat)
    int sTThang; //So tran thang
    int sTHoa; //So tran hoa
    int sTTThua; //So tran thua
    int sBThang; //So ban thang
    int sBThua; //So ban thua
    int hieuSo; //Hieu so ban thang thua
    int diem; //Diem
    int xh; //Xep hang
    data2 x;
    in >> tenDoi; strcpy_s(x.tenDoi, 15, tenDoi);
    in >> st; x.st = st;
    in >> sTThang; x.sTThang = sTThang;
    in >> sTHoa; x.sTHoa = sTHoa;
    in >> sTTThua; x.sTTThua = sTTThua;
    in >> sBThang; x.sBThang = sBThang;
    in >> sBThua; x.sBThua = sBThua;
    in >> hieuSo; x.hieuSo = hieuSo;
    in >> diem; x.diem = diem;
    in >> xh; x.xh = xh;
    InsertTail2(h, x);
    while (!in.eof())
    {
        in >> tenDoi; strcpy_s(x.tenDoi, 15, tenDoi);
        in >> st; x.st = st;
        in >> sTThang; x.sTThang = sTThang;
        in >> sTHoa; x.sTHoa = sTHoa;
        in >> sTTThua; x.sTTThua = sTTThua;
        in >> sBThang; x.sBThang = sBThang;
        in >> sBThua; x.sBThua = sBThua;
        in >> hieuSo; x.hieuSo = hieuSo;
        in >> diem; x.diem = diem;
        in >> xh; x.xh = xh;

        InsertTail2(h, x);
    }
    in.close();
}
//Het vong j : con tro tro den tran sau tran cuoi vong j
NODE1 *Tro_Sau_Cuoi_Vong(LIST_KQTD l, int vong)
{
    NODE1 *p;
    int sTran = 10 * vong;
    int i = 0;
    for (p = l.pHead; p != NULL; p = p->pNext)
    {
        i++;
    }
}
```

```

        if (i > sTran)
            break;
    }
    return p;
}
//Tao list ket qua cac doi den vong tu list ket qua tran dau cua ca doi den vong
//Xet tung doi, duyet qua ket qua cac tran, kiem tra trung ten doi : tinh cac gia tri du lieu thanh phan
void TaoList_KQDB_Vong(LIST_KQXH &h, LIST_KQTD l, int vong)
{
    NODE1 *p; //duyet tran i
    NODE2 *q; // duyet doi j
    data2 x; //ket qua doi bong
    NODE1 *r = Tro_Sau_Cuoi_Vong(l, vong);
    KhoiTao_List_KQXH(h);
    for (q = h.pHead; q != NULL; q = q->pNext) //Voi moi doi q
    {
        x = q->kqxh; // doi x
        for (p = l.pHead; p != r; p = p->pNext) //duyet den tran cuoi cung vong muon xep hang
        {
            if (_strcmpi(x.tenDoi, p->kqtd.cNha) == 0) //tinh du lieu lien quan khi la chu nha
            {
                x.sBThang += p->kqtd.bTChu;
                x.sBThua += p->kqtd.bTKhach;
                if (p->kqtd.bTChu > p->kqtd.bTKhach)
                {
                    x.sTThang++;
                    x.diem += 3;
                }
                else
                if (p->kqtd.bTChu < p->kqtd.bTKhach)
                    x.sTThua++;
                else
                {
                    x.sTHoa++;
                    x.diem += 1;
                }
            }
            else
            if (_strcmpi(x.tenDoi, p->kqtd.khach) == 0) //tinh du lieu lien quan khi la chu nha
            {
                x.sBThang += p->kqtd.bTKhach;
                x.sBThua += p->kqtd.bTChu;
                if (p->kqtd.bTChu < p->kqtd.bTKhach)
                {
                    x.sTThang++;
                    x.diem += 3;
                }
                else
                if (p->kqtd.bTChu > p->kqtd.bTKhach)
                    x.sTThua++;
                else
                {
                    x.sTHoa++;
                    x.diem += 1;
                }
            }
            x.hieuSo = x.sBThang - x.sBThua;
            x.st = x.sTThang + x.sTHoa + x.sTThua;
        }
    }
}

```

```

        q->kqhx = x;
    }
}

```

4. Tập tin **04_XepHang.h** :

//Thu vien nay tien hanh danh so xep hang dua vao bang ket qua cac doi:

//Dau tien, Bang ket qua cac doi se sap giam theo thu tu: diem, hieu so, so ban thang; tang theo ten doi - Chua danh so xep hang

//Sau do tien hanh danh so xep hang (xuất phát từ 1, số thứ tự tăng dần đến hết các đối),

//cac truong hop trung nhau ve cac tham so xep loai thi cung hang.

```

//////////
//          Xep hang cac doi          //
//////////

```

```

void HoanVi(data2 &x, data2 &y)
{

```

```

    data2 t = x;
    x = y;
    y = t;
}

```

//Sap ket qua giam theo diem, theo hieu so, theo so ban thang, tang theo ten doi. Chua danh so xep hang

```

void SapGiamTheoDiem_HieuSo_SoBT_DenVong(LIST_KQXH &h, int vong)
{

```

```

    NODE2 *p, *q; //Duyet doi bong

```

```

    //Sap giam theo diem

```

```

    for (p = h.pHead; p != h.pTail; p = p->pNext)
        for (q = p->pNext; q != NULL; q = q->pNext)
            if (p->kqhx.diem < q->kqhx.diem)
                HoanVi(p->kqhx, q->kqhx);

```

//Sap bang ket qua giam theo tong diem, giam theo hieu so

```

    for (p = h.pHead; p != h.pTail; p = p->pNext)
        for (q = p->pNext; q != NULL; q = q->pNext)
            if (p->kqhx.diem == q->kqhx.diem)
                if (p->kqhx.hieuSo < q->kqhx.hieuSo)
                    HoanVi(p->kqhx, q->kqhx);

```

//Sap bang ket qua giam theo tong diem, giam theo hieu so, giam theo so ban thang

```

    for (p = h.pHead; p != h.pTail; p = p->pNext)
        for (q = p->pNext; q != NULL; q = q->pNext)
            if (p->kqhx.diem == q->kqhx.diem)
                if (p->kqhx.hieuSo == q->kqhx.hieuSo)
                    if (p->kqhx.sBThang < q->kqhx.sBThang)
                        HoanVi(p->kqhx, q->kqhx);

```

//Sap bang ket qua giam theo tong diem, giam theo hieu so, giam theo so ban thang, tang theo ten doi

```

    for (p = h.pHead; p != h.pTail; p = p->pNext)
        for (q = p->pNext; q != NULL; q = q->pNext)
            if (p->kqhx.diem == q->kqhx.diem)
                if (p->kqhx.hieuSo == q->kqhx.hieuSo)
                    if (p->kqhx.sBThang == q->kqhx.sBThang)
                        if (_strcmpi(p->kqhx.tenDoi, q->kqhx.tenDoi) > 0)
                            HoanVi(p->kqhx, q->kqhx);

```

```

}

```

//Danh so xep hang, thu tu khong lien tục 1 đến sd, mà nhảy theo số đối bằng nhau về các tham số xếp loại

```

void XepHang_DenVong(LIST_KQXH &h, int vong)
{

```

```

    int vt = 1, //luu thu hang, dau tien la 1

```

```

        soVT; //so luong cac doi cung 1 hang, khi dang xet mot hang

```

```

    NODE2 *p, *q; //duyet doi

```

```

    SapGiamTheoDiem_HieuSo_SoBT_DenVong(h, vong);

```

//Bang ket qua h da giam theo diem, hieu so, so ban thang va tang theo ten doi

```

    p = h.pHead;

```

```

    p->kqhx.xh = 1; //doi dau tien xep hang 1

```

```

soVT = 1; //khởi tạo số đối bằng nhau là 1, khi đang tính các đối bằng nhau
for (q = p->pNext; q != NULL; q = q->pNext)
{
    if (q->kq.xh.diem == p->kq.xh.diem &&
        q->kq.xh.hieuSo == p->kq.xh.hieuSo &&
        q->kq.xh.sB.Thang == p->kq.xh.sB.Thang)
    {
        q->kq.xh.xh = vt; //xếp các đối bằng nhau cùng hàng
        soVT++; //số đối bằng nhau tăng thêm 1
    }
    else
    {
        vt += soVT; //hàng đối kế tiếp tăng thêm soVT, (không chắc là 1)
        q->kq.xh.xh = vt;
        soVT = 1; //khởi tạo lại soVT = 1
    }
    p = q; //khởi tạo lại p
}

//Xuat bang ket qua xep hang cac doi
//
//
//
//Xuat tieu de bang ket qua
void XuatTieuDe_KQ()
{
    cout << endl;
    cout << ":\n";
    for (int i = 1; i <= 70; i++)
        cout << NGANGDOI;
    cout << ":\n";
    cout << endl;
    cout << setiosflags(ios::left)
    << ":\n";
    << setw(15) << "Ten doi"
    << ":\n";
    << setw(4) << "ST"
    << ":\n";
    << setw(4) << "T" //So tran thang
    << ":\n";
    << setw(4) << "H" //So tran Hoa
    << ":\n";
    << setw(4) << "B" //So tran thua
    << ":\n";
    << setw(4) << "BT" //So ban thang
    << ":\n";
    << setw(4) << "BB" //So ban thua
    << ":\n";
    << setw(4) << "HS" //hieu so
    << ":\n";
    << setw(4) << "Diem" //Diem
    << ":\n";
    << setw(4) << "XH" //Xep hang
    << ":\n";
    cout << endl;
    cout << ":\n";
    for (int i = 1; i <= 70; i++)
        cout << NGANGDOI;
    cout << ":\n";
}

```

```
//Xuat KQ 1 doi
void XuatKQ_1D(data2 x)
{
    cout << setiosflags(ios::left)
        << ":\n"
        << setw(15) << x.tenDoi
        << ":\n"
        << setw(4) << x.st
        << ":\n"
        << setw(4) << x.sTThang//So tran thang
        << ":\n"
        << setw(4) << x.sTHoa//So tran Hoa
        << ":\n"
        << setw(4) << x.sTThua//So tran thua
        << ":\n"
        << setw(4) << x.sBThang//So ban thang
        << ":\n"
        << setw(4) << x.sBThua//So ban thua
        << ":\n"
        << setw(4) << x.hieuSo//hieu so
        << ":\n"
        << setw(4) << x.diem//Diem
        << ":\n"
        << setw(4) << x.xh//Xep hang
        << ":\n";
}

//Xuat bang ket qua xep hang
void Xuat_KQXH(LIST_KQXH h)
{
    NODE2 *p;
    XuatTieuDe_KQ();
    for (p = h.pHead; p != h.pTail; p = p->pNext)
    {
        cout << endl;
        XuatKQ_1D(p->kqxh);
        cout << endl;
        cout << ":\n";
        for (int i = 1; i <= 70; i++)
            cout << NGANGDON;
        cout << ":\n";
    }
    cout << endl;
    XuatKQ_1D(p->kqxh);
    cout << endl;
    cout << ":\n";
    for (int i = 1; i <= 70; i++)
        cout << NGANGDOI;
    cout << ":\n";
}
```

5. Tập tin dữ liệu **kqtd.txt** :

- Lưu trữ kết quả các trận đấu (theo mỗi vòng)
 - o các trận đấu chưa đấu, tỉ số trận đấu khởi tạo là -1,-1.
 - o Các giá trị thuộc tính khác nhau tách biệt bằng các ký tự tách : khoảng trắng, tab, xuống hàng
 - o Tên các đội đôi một khác nhau, có nhiều từ thì phải kết nối bằng dấu '_'.

Nội dung kết quả trận đấu của 2 vòng đầu :

Thực hành cấu trúc dữ liệu và thuật giải_2019-2020_HK2

1	13-08-16	18:30	Hull_City	2	1	Leicester
1	13-08-16	21:00	Southampton	1	1	Watford
1	13-08-16	21:00	Middlesbrough	1	1	Stoke_City
1	13-08-16	21:00	Everton	1	1	Tottenham
1	13-08-16	21:00	Crystal_Palace	0	1	West_bromwich
1	13-08-16	21:00	Burley	0	1	Swansea_City
1	13-08-16	23:30	MC	2	1	Sunderland
1	14-08-16	19:30	Bournemouth	1	3	MU
1	14-08-16	22:00	Arsenal	3	4	Liverpool
1	16-08-16	02:00	Chelsea	2	1	West_Ham
2	20-08-16	02:00	MU	2	0	Southampton
2	20-08-16	18:30	Stoke_City	1	4	MC
2	20-08-16	21:00	West_bromwich	1	2	Everton
2	20-08-16	21:00	Watford	1	2	Chelsea
2	20-08-16	21:00	Tottenham	1	0	Crystal_Palace
2	20-08-16	21:00	Swansea_City	0	2	Hull_City
2	20-08-16	21:00	Burley	2	0	Liverpool
2	20-08-16	23:30	Leicester	0	0	Arsenal
2	21-08-16	19:30	Sunderland	1	2	Middlesbrough
2	21-08-16	22:00	West_Ham	1	0	Bournemouth

// các vòng khác ...

6. Tập tin dữ liệu **Khoitao_KQDB.txt** :

- Lưu trữ khởi tạo kết quả các đội bóng : các trường (ngoài tên các đội bóng) đều khởi tạo bằng 0.

Arsenal	0	0	0	0	0	0	0	0	0
Bournemouth	0	0	0	0	0	0	0	0	0
Burley	0	0	0	0	0	0	0	0	0
Chelsea	0	0	0	0	0	0	0	0	0
Crystal_Palace	0	0	0	0	0	0	0	0	0
Everton	0	0	0	0	0	0	0	0	0
Hull_City	0	0	0	0	0	0	0	0	0
Leicester	0	0	0	0	0	0	0	0	0
Liverpool	0	0	0	0	0	0	0	0	0
MC	0	0	0	0	0	0	0	0	0
Middlesbrough	0	0	0	0	0	0	0	0	0
MU	0	0	0	0	0	0	0	0	0
Southampton	0	0	0	0	0	0	0	0	0
Stoke_City	0	0	0	0	0	0	0	0	0
Sunderland	0	0	0	0	0	0	0	0	0
Swansea_City	0	0	0	0	0	0	0	0	0
Tottenham	0	0	0	0	0	0	0	0	0
Watford	0	0	0	0	0	0	0	0	0
West_bromwich	0	0	0	0	0	0	0	0	0
West_Ham	0	0	0	0	0	0	0	0	0

LAB 05. Danh sách liên kết đơn : Các phép toán tập hợp

A. Mục tiêu

- Tìm hiểu các phép toán tập hợp.
- Cài đặt các phép toán tập hợp, với tập hợp cài đặt bằng DSLK đơn

B. Yêu cầu

- Phần ôn tập (C) : sinh viên tự ôn.
- **Buổi thực hành 6** (2 tiết đầu / 3 t)
 - Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Bài 1 E (bài tập)
 - Yêu cầu lưu trữ
 - ✓ Tạo thư mục **MaSV_HoVaTen_Lab05** chứa trong ổ đĩa qui định.
 - ✓ Tạo project lưu trữ trong thư mục trên.
 - ✓ Xóa thư mục debug trong project.

C. Ôn tập

a. Các phép toán cơ bản trên tập hợp:

1. Giao 2 tập hợp : (tập A giao B)
 $A \cap B = \{x : x \in A \wedge x \in B\}$
2. Hợp 2 tập hợp : (tập A hợp B)
 $A \cup B = \{x : x \in A \vee x \in B\}$
3. Tập hiệu : (tập A hiệu B)
 $A \setminus B = \{x : x \in A \wedge x \notin B\}$
4. Hiệu đối xứng : (tập A hiệu đối xứng B)
 $A \Delta B = (A \setminus B) \cup (B \setminus A)$
5. Tích Descartes : (tập tích Descartes của A và B)
 $A \times B = \{(a,b) : a \in A, b \in B\}$

Lưu ý : Các phần tử trong tập hợp là đôi một khác nhau.

b. Ứng dụng danh sách liên kết đơn cài đặt tập hợp.

<ul style="list-style-type: none"> • Tập hợp là kiểu danh sách liên kết đơn, đặt tên là : SET • Mỗi phần tử của tập hợp là một nút, kiểu của nút này được đặt tên là NODE. • Mỗi nút này có 2 thành phần, một thành phần chứa dữ liệu của nút được đặt tên info có kiểu data, thành phần thứ 2 là con trỏ, trỏ đến nút kế tiếp. 	<ul style="list-style-type: none"> • Kiểu của thành phần dữ liệu của các nút trong tập hợp (Giả sử là kiểu int): <code>typedef int data;</code> • Kiểu Phần tử (Nút) của tập hợp (DSLK): - Định nghĩa ban đầu <pre>struct tagNode { data info; tagNode* pNext; };</pre> - Đổi lại tên: <code>typedef tagNode NODE;</code> • Kiểu tập hợp SET (DSLK đơn): <code>struct SET</code> <pre>{ NODE* pHead; NODE* pTail; };</pre>
--	--

D. Mô tả các phép toán trên tập hợp

1. Giao hai tập hợp :

Input: A,B;

Output: $C = A \cap B$;

Giao(SET A, SET B) \equiv

```
SET C;  
Tạo tập C rỗng;  
NODE *p = A.pHead;  
Trong khi (p !=NULL)  
{  
    Nếu ( p->info  $\in$  B)  
        Chèn p->info vào cuối C;  
    p = p -> pNext;  
}  
Return C;
```

2. Hợp hai tập hợp :

Input: A,B;

Output: $A \cup B = C$;

Hop(SET A, SET B) \equiv

```
SET C;  
C = A;  
NODE *p = B.pHead;  
Trong khi (p !=NULL)  
{  
    Nếu (p->Info  $\notin$  A) // không thuộc  
        Chèn p->info vào cuối C;  
    p = p -> pNext;  
}  
return C;
```

3. Hiệu 2 tập hợp (A hiệu B)

Input: A,B;

Output: $C = A \setminus B$;

Hieu(SET A, SET B) \equiv

```
SET C;  
Tạo tập C rỗng;  
NODE *p = A.pHead;  
Trong khi (p !=NULL) //Duyệt A  
{  
    Nếu (p->info  $\notin$  B) // không thuộc  
        Chèn p->info vào cuối C;  
    p = p -> pNext;  
}  
return C;
```

4. Hiệu đối xứng:

Input: A,B;

Output: $C = A \Delta B$;

Hieu(SET A, SET B) \equiv

```
SET C, D, E;  
D =  $A \setminus B$ ;  
E =  $B \setminus A$ ;  
C =  $D \cup E$ ;  
return C;
```


5. Tích Descartes:

Input: A,B;

Output: C = AxB ;

Des(SET A, SET B) ≡

```

    Nếu (A.pHead == NULL || B.pHead == NULL)
        return;
    Xuất ("A x B = {");
    for (p = A.pHead; p != NULL; p = p->pNext)
        for (q = B.pHead; q != NULL; q = q->pNext)
        {
            Xuất (p->info, q->info);
            if (p != A.pTail || q != B.pTail)
                Xuất ',';
        }
    Xuất '}';

```

6. Kiểm tra phần tử thuộc tập hợp:

Input: A, x;

Output: 1; x ∈ A

0; Ngược lại

Thuoc(SET A, data x) ≡

```

    NODE *p = A.pHead;
    Trong khi (p != NULL)
    {
        if (p->Info == x )
            return 1;
        p = p -> pNext;
    }
    return 0;

```

7. Bao hàm trong, chứa trong : Kiểm tra A có bao hàm trong (chứa trong) B ($A \subseteq B$)

(B chứa A, B bao hàm A, $B \supseteq A$)

Input: A,B

Output: 1; Nếu $A \subseteq B$

0; Ngược lại

BaoHam(SET A, SET B) ≡

```

    NODE *p = A.pHead;
    Trong khi (p != NULL) ?? Duyệt A
    {
        Nếu (p->info ∉ B) // không thuộc
            return 0;
        p = p -> pNext;
    }
    //  $A \subseteq B$ 
    return 1;
}

```

8. Quan hệ bằng nhau

Input: A,B

Output: 1; Nếu A==B

0; Ngược lại

BangNhau(SET A, SET B) ≡

```

    Nếu (A ⊆ B) && (B ⊆ A))
        return 1;
    return 0;

```

9. Lực lượng tập hợp

//Trả về số phần tử của tập hợp (DSLK đơn)

10. Nhập dữ liệu cho tập hợp

Sử dụng hàm nhập dữ liệu trong DSLK đơn (từ bàn phím hay từ tập tin văn bản)
(Kiểm tra các phần tử trong tập hợp phải đôi một khác nhau)

11. Xuất dữ liệu của tập hợp ra màn hình

Sử dụng hàm xuất dữ liệu trong DSLK đơn., xuất ra dạng : {9, 4, 5,...}

E. Bài tập

Bài 1:

Viết chương trình tùy chọn cài đặt các phép toán tập hợp, dữ liệu của tập hợp là số nguyên:

1. Nhập dữ liệu cho tập hợp
2. Xem dữ liệu tập hợp
3. Giao 2 tập hợp
4. Hợp 2 tập hợp
5. Hiệu 2 tập hợp
6. Hiệu đối xứng 2 tập hợp
7. Tích Descartes 2 tập hợp
8. Kiểm tra phần tử có thuộc vào tập hợp
9. Kiểm tra quan hệ bao hàm giữa 2 tập
10. Tính lực lượng tập hợp

Dữ liệu cho trong các tập tin :

- **"A.txt" :**

9 7 12 6 3 0 5 8 1 4
//A = {9, 7, 12, 6, 3, 0, 5, 8, 1, 4}

- **"B.txt" :**

10 2 8 7 16 4
//B = {10, 2, 8, 7, 16, 4}

LAB 05. Danh sách liên kết đơn : Các phép toán trên đa thức rời rạc

A. Mục tiêu

- Tìm hiểu đa thức rời rạc và các phép toán trên đa thức rời rạc.
- Cài đặt các phép toán đa thức, với đa thức rời rạc cài đặt bằng DSLK đơn

B. Yêu cầu

- Phần ôn tập (C) : sinh viên tự ôn.

- **Buổi thực hành 6** (1 tiết cuối / 3 t)

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Bài 1E (bài tập)
 - Yêu cầu lưu trữ
 - ✓ Trong thư mục **MaSV_HoVaTen_Lab05** tạo project tương ứng
 - ✓ Xóa thư mục debug trong project
 - ✓ Nén thư mục
 - Hình thức thu bài
 - ✓ Hệ thống thu bài tại phòng lab.

C. Ôn tập

I. Đa thức:

Đa thức A một biến theo x, hệ số thực, bậc n có dạng:

$$A(x) = \sum_{i=0}^n a_i x^i = a_0 + a_1 x + \dots + a_n x^n; a_n \neq 0; a_i \in R, \forall i = 0 \dots n$$

Ta xét các đa thức trong đó chỉ có một số các hệ số khác 0, các đa thức như vậy ta gọi là đa thức rời rạc. Các đa thức rời rạc có dạng :

$$A(x) = \sum_{j=0}^m a_{i_j} x^{i_j} = a_{i_0} x^{i_0} + a_{i_1} x^{i_1} + \dots + a_{i_m} x^{i_m}; i_j \in N, a_{i_j} \neq 0; a_{i_j} \in R, \forall j = 0 \dots m$$

Khi lưu trữ các đa thức rời rạc, ta cần lưu trữ các hệ số và số mũ tương ứng.

II.. Ứng dụng danh sách liên kết đơn cài đặt đa thức rời rạc

<ul style="list-style-type: none"> - Các hệ số trong đa thức rời rạc đều khác 0. - Khi lưu trữ, ta lưu trữ 2 giá trị cho mỗi hạng tử đa thức : hệ số và số mũ tương ứng • Đa thức rời rạc là DSLK đơn, đặt tên là : POLYNOME • Các phần tử (nút) của đa thức có cùng kiểu dữ liệu, kiểu của nút này được đặt tên là NODE. • Mỗi nút có 2 thành phần, một thành phần chứa dữ liệu của nút được đặt tên là info có kiểu data, thành phần thứ 2 là con trỏ, trỏ đến nút kế tiếp. • data là kiểu cấu trúc có 2 trường dữ liệu mô tả đặc trưng của một đa thức, bao gồm: <ul style="list-style-type: none"> - Hệ số của đa thức (khác 0) - Số mũ 	<ul style="list-style-type: none"> • Kiểu của thành phần dữ liệu của các nút: <pre>struct poly { double coef; int expo; }</pre> typedef poly data; • Kiểu Phần tử (Nút) của Đa thức <pre>struct tagNode { data info; tagNode* pNext; };</pre> // Đổi lại tên: typedef tagNode NODE; • Kiểu Đa thức (DSLK đơn): <pre>struct POLYNOME { NODE* pHead; NODE* pTail; };</pre>
---	--

III. Các thao tác cơ bản trên đa thức rời rạc :

1. Tạo nút mới :

```
NODE* GetNode(data x);
```

2. Tạo đa thức rỗng

```
void CreatPoly(POLYNOME &A);
```

3. Chèn x vào sau đa thức rời rạc :

```
void InsertTail(POLYNOME &A, data x);
```

//....

Lưu ý là trong đa thức rời rạc, chỉ lưu trữ các hệ số khác 0.

D. Mô tả các phép toán trên đa thức rời rạc

1. Nhập một đa thức (chuyển từ tập tin sang đa thức : DSLK đơn)

- Input f
- Output A
- Nguyên mẫu:

```
void File_Poly(char *f, POLYNOME &A)
```


Hoặc :

```
int File_Poly(char *f, POLYNOME &A)
```
- Tập tin đa thức f có nhiều hàng, mỗi hàng gồm 2 số hạng (cách nhau ít nhất 1 khoảng trắng) : số hạng đầu là một số thực khác 0 (hệ số đa thức), số hạng sau là một số nguyên không âm (số mũ của đa thức).
- Kết quả : A biểu diễn dạng rút gọn và số hạng tăng dần theo số mũ.

2. Xuất đa thức theo định dạng:

- Input A (dạng rút gọn theo nghĩa : số mũ của các số hạng đôi một khác nhau, và tăng dần)
- Output : xuất A theo định dạng đa thức có số mũ tăng dần, các hệ số khác 0. Nếu hệ số bằng 1 hay bằng -1 thì không hiển thị 1; nếu số mũ bằng 1 thì cũng không hiển thị số mũ; dùng dấu ^ để biểu diễn ký hiệu mũ
 Chẳng hạn : $A(x) = 2 + 3x - 7x^5 + 4x^6 - x^8$
- Nguyên mẫu:

```
void Output_Poly(POLYNOME A);
```

//He so luon khac 0 nen khong can kiem tra dieu kien nay

```
void Output_Poly(POLYNOME A)
{
    if (A.pHead == NULL)
    {
        cout << "\nDa thuc rong";

        return;
    }
    NODE *p;
    //xu ly nhan tu dau tien, so mu co the == 0
    p = A.pHead;
    if (p->info.expo == 0) //so mu = 0
        cout << p->info.coef;
    else // so mu > 0 : >= 1
    {
        if (p->info.expo == 1) // so mu = 1
        {
            if (p->info.coef == 1) //he so == 1
                cout << "x ";
            else // he so !=1
            {
                if (p->info.coef == -1)
                    cout << "-x ";
                else
                    cout << p->info.coef << "x ";
            }
        }
        else //so mu > 1
        {
            if (p->info.coef == 1) //he so == 1
                cout << "x^" << p->info.expo << " ";
            else // he so !=1
            {
                if (p->info.coef == -1)
                    cout << "-x^" << p->info.expo << " ";
                else
                    cout << p->info.coef << "x^" << p->info.expo << " ";
            }
        }
    }
    //xu ly cac hang tu tiep theo
    p = p->pNext;
    while (p != NULL)
    {
        //so mu > 0
        if (p->info.expo == 1) // so mu = 1
        {
            if (p->info.coef == 1) //he so == 1
                cout << "+x ";
            else // he so !=1
            {
                if (p->info.coef == -1)
                    cout << "-x ";
                else
                    if (p->info.coef > 0)
                        cout << "+" << p->info.coef << "x ";
                    else
                        cout << p->info.coef << "x ";
            }
        }
        else //so mu > 1
        {
            if (p->info.coef == 1) //he so == 1
                cout << "+x^" << p->info.expo << " ";
        }
    }
}
```

```

else // he so !=1
    if (p->info.coef == -1)
        cout << "-x^" << p->info.expo << " ";
    else
        if (p->info.coef > 0)
            cout << "+" << p->info.coef << "x^" << p->info.expo << " ";
        else //(he so < 0)
            cout << p->info.coef << "x^" << p->info.expo << " ";
    }
    p = p->pNext;
}
}

```

3. Sắp xếp đa thức tăng theo số mũ.

- Input: A
- Output: đa thức được sắp tăng theo mũ
- Nguyên mẫu: void Sort_Expo(POLYNOME &A);

Sort_Expo(A) =
 //Có thể sử dụng các thuật giải sắp trực tiếp
 //trường khóa là số mũ
 //hoán đổi 2 cấu trúc kiểu data

4. Rút gọn một đa thức.

Ví dụ: $A(x) = 4x^2 + 6x^4 - 7x^2 = -3x^2 + 6x^4$

- Input A
- Output A
- Nguyên mẫu : void Reduction (POLYNOME &A) ;

```

void Reduction(POLYNOME &A)
{
    if (A.pHead == NULL)
    {
        cout << "\nDa thuc rong";
        system("PAUSE");
        return;
    }

    int Kq;
    POLYNOME B;
    NODE *pB;
    NODE *pA;
    pA = A.pHead;
    CreatPoly(B);

    while (pA != NULL)
    {
        Kq = 0;
        pB = B.pHead;
        while (pB != NULL)
        {
            if (pB->info.expo == pA->info.expo)
            {
                Kq = 1;
                break;
            }
            pB = pB->pNext;
        }
        if (Kq == 1)
    }
}

```

```

        pB->info.coef += pA->info.coef;
    else
        InsertTail(B, pA->info);
    pA = pA->pNext;
}
A = B;
}

```

5. Tổng của 2 đa thức.

- Input: A, B
- Output: C = A+B
- Nguyên mẫu: POLYNOME ADD(POLYNOME A, POLYNOME B);

```

Add(A,B) ≡
    POLYNOME C;
    data x;
    Khởi tạo C rỗng;
    pA = A.pHead; pB = B.pHead;
    Trong khi (pA != NULL && pB != NULL)
    {
        Nếu (pA->info.expo < pB->info.expo)
        {
            Chèn pA->info vào cuối C;
            pA = pA->pNext ;
        }

        Nếu (pA->info.expo > pB->info.expo)
        {
            Chèn pB->info vào cuối C;
            pB = pB->pNext ;
        }

        Nếu (pA->info.expo == pB->info.expo)
        {
            x.coef = pA->info.coef + pB->info.coef ;
            x.expo = pA->info.expo ;
            Nếu (x.coef != 0)
                Chèn x vào cuối C;
            pA = pA->pNext ;
            pB = pB->pNext ;
        }
    }

    Trong khi (pA != NULL)
    {
        Chèn pA->info vào cuối C;
        pA = pA->pNext ;
    }

    Trong khi (pB != NULL)
    {
        Chèn pB->info vào cuối C;
        pB = pB->pNext ;
    }
    Reduction(C);
    Sort_Expo(C);
    return C ;

```

6. Hiệu của 2 đa thức.

- Input: A,B
- Output: C = A-B

```

POLYNOME Minus(POLYNOME A, POLYNOME B)
{
    POLYNOME C;
    NODE *pA, *pB;
    data x;
    CreatPoly(C);
    pA = A.pHead;
    pB = B.pHead;
    while (pA != NULL && pB != NULL)
    {
        if (pA->info.expo < pB->info.expo)
        {
            InsertTail(C, pA->info);
            pA = pA->pNext;
        }
        else
        {
            if (pB->info.expo < pA->info.expo)
            {
                x.coef = -pB->info.coef;
                x.expo = pB->info.expo;

                InsertTail(C, x);
                pB = pB->pNext;
            }
            else //if (pA->info.expo == pB->info.expo)
            {
                x.coef = pA->info.coef - pB->info.coef;
                x.expo = pA->info.expo;
                if(x.coef != 0)
                    InsertTail(C, x);
                pA = pA->pNext;
                pB = pB->pNext;
            }
        }
    }

    while (pA != NULL)
    {
        InsertTail(C, pA->info);
        pA = pA->pNext;
    }

    while (pB != NULL)
    {
        x.coef = -pB->info.coef;
        x.expo = pB->info.expo;
        InsertTail(C, x);
        pB = pB->pNext;
    }

    Reduction(C);
    Sort_expo(C);
    return C;
}

```

7. Nhân một phân tử với đa thức.

- input: A, $a_i x^i$ ($i = p$)
- Output: $B = a_i x^i * A$ // $B = p * A$

// $a_i = p \rightarrow \text{info.Coeff}$; $i == p \rightarrow \text{info.Expo}$

- Nguyên mẫu: `POLYNOME Mult_Node(POLYNOME A, NODE *p);`

```

Mult_Node (A, p) ≡
    POLYNOME B;
    data t;
    Khởi tạo B rỗng
    pA = A.pHead;
    Trong khi (pA != NULL)
    {
        t.Expo = pA->Info.Expo + p->Info.Expo;
        t.Coeff = pA->Info.Coeff * p->Info.Coeff;
        Chèn t vào cuối B
        pA = pA -> pNext;
    }
    Reduction(B);
    Sort_Expo(B);
    return B;
    
```

8. Nhân hai đa thức.

- Input: A,B
- Output: $C = AB$

```

POLYNOME Mult_Poly(POLYNOME A, POLYNOME B)
{
    POLYNOME C, T1, T2;
    CreatPoly(C);
    CreatPoly(T1);
    CreatPoly(T2);

    NODE *pA, *p;
    pA = A.pHead;
    while (pA != NULL)
    {
        T1 = Mult_Node(B, pA);
        T2 = ADD(C, T1);
        C = T2;
        pA = pA->pNext;
    }
    return C;
}
    
```

9. Tính giá trị đa thức $A(x)$, với giá trị x được nhập từ bàn phím.

- Input: A, x
- Output: $A(x)$
- Nguyên mẫu: `double ValueP(POLYNOME A, double x)`

```

ValueP (A, x) ≡
    double V = 0, t;
    pA = A.pHead ;
    Trong khi (pA != NULL)
    {
        t = pA->info.coef * x^(pA->info.expo) ;
        V = V + t;
        pA = pA -> Next;
    }
    return V;
    
```

10. Tính đạo hàm của đa thức.

- Input: A
- Output: A'

POLYNOME DeriveA(POLYNOME A)

```
{
    POLYNOME B;
    CreatPoly(B);
    NODE *pA = A.pHead;
    data x;
    while (pA != NULL)
    {
        if (pA->info.expo == 0)
        {
            x.coef = 0;
            x.expo = 0;
        }
        else
        {
            x.coef = pA->info.expo * pA->info.coef;
            x.expo = pA->info.expo - 1;
        }
        if (x.coef != 0)
            InsertTail(B, x);
        pA = pA->pNext;
    }
    Reduction(B);
    Sort_Expo(B);
    return B;
}
```

E. Bài tập

Bài 1:

Viết chương trình tùy chọn cài đặt các phép toán trên đa thức rồi rạc:

11. Nhập dữ liệu cho đa thức
12. Xem đa thức
13. Tổng 2 đa thức
14. Hiệu 2 đa thức
15. Tích 2 đa thức
16. Tính đạo hàm đa thức
17. Tính tích phân hàm đa thức
18. Tính giá trị đa thức tại x

Dữ liệu các đa thức rời rạc cho trong các tập tin sau, theo định dạng trên mỗi hàng có 2 số, số thực đứng trước là hệ số của hạng tử đa thức, số nguyên không âm đứng sau là số mũ tương ứng:

- "A.txt" :

-1 0 //-1x⁰ : -1

2 2 //2x²

1 6 //1x⁶

//A(x) = -1 + 2x² + x⁶

// Biểu diễn dạng : A(x) = -1 + 2x² + x⁶

- "B.txt" :

3 1

-1 5

1 6

3 8

//B(x) = 3x - x⁵ + x⁶

//B(x) = 3x - x⁵ + x⁶ + 3x⁸

LAB 06. Danh sách liên kết đơn : Ngăn xếp (Stack)

A. Mục tiêu

- Thực hiện các thao tác cơ bản trên STACK tổ chức bằng danh sách liên kết đơn.
- Cài đặt một số ứng dụng trên STACK

B. Yêu cầu

- Phần ôn tập (C) : sinh viên tự ôn.
- **Buổi thực hành 7** (2 tiết đầu / 3 t)
 - Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : **Bài 4,5 D (luyện tập)**
 - Yêu cầu lưu trữ
 - ✓ Tạo thư mục **MaSV_HoVaTen_Lab06** chứa trong ổ đĩa qui định.
 - ✓ Mỗi bài, tạo project lưu trữ trong thư mục trên.
 - ✓ Xóa thư mục debug trong mỗi project.

C. Ôn tập

I. Cách tổ chức kiểu dữ liệu STACK bằng DSLK đơn

<ul style="list-style-type: none">• STACK (Ngăn xếp) là kiểu dữ liệu tuyến tính nhằm biểu diễn các đối tượng được xử lý theo kiểu "vào sau ra trước" (LIFO: Last In, First Out).• Dùng danh sách liên kết đơn để biểu diễn STACK• Các phép toán thêm vào và lấy ra được thực hiện cùng ở một đầu danh sách (gọi là đỉnh của STACK).	<ul style="list-style-type: none">• Kiểu của thành phần dữ liệu của các nút trong stack (giả sử là kiểu int): <code>typedef int data;</code>• Kiểu Phần tử (Nút) của tập hợp (DSLK): - Định nghĩa ban đầu <pre>struct tagNode { data info; tagNode* pNext; };</pre> - Đổi lại tên: <code>typedef tagNode NODE;</code>• Kiểu dữ liệu STACK: <pre>struct STACK { NODE* pHead; NODE* pTail; };</pre>
---	--

II. Các thao tác cơ bản trên Stack

1. Tạo nút mới: `GetNode (x)` :

- Chức năng: Tạo ra một nút của Sack với thành phần dữ liệu là x (Thành phần liên kết của nút mới là con trỏ có giá trị NULL)
- Input : x
- Output : trả về con trỏ - lưu trữ địa chỉ của pt vừa tạo (nếu tạo thành công) - có giá trị NULL (nếu ngược lại).
- Nguyên mẫu của hàm: `NODE* GetNode(data x);`

2. Khởi tạo Stack rỗng: `CreatStack (s)`

- Chức năng: Tạo ra một Stack s rỗng.
- Input : s

- Output :s
 - Nguyên mẫu của hàm: void CreatStack(STACK &s);
3. Thêm một giá trị vào đỉnh stack (đầu DSLK đơn)
- Chức năng: Tạo nút với dữ liệu x, chèn nút này vào đầu stack,
 - Input : x , s
 - Output : s
 - Nguyên mẫu của hàm: **void Push(STACK &s, data x);**
4. Lấy nút ở đỉnh stack (Hủy nút đầu DSLK đơn)
- Chức năng: Hủy nút ở đỉnh (C/K trả về dữ liệu của đỉnh Stack)
 - Input : s
 - Output :x (/s)
 - Nguyên mẫu của hàm: **data Pop(STACK &s);**
void Pop(STACK &s);
5. Xem thông tin ở đỉnh stack (xuất dữ liệu nút đầu, không hủy)
- Chức năng: Xem thông tin ở đỉnh stack
 - Input : s
 - Output : x //dữ liệu đỉnh stack
 - Nguyên mẫu của hàm: **data Top(STACK s);**
6. Nhập dữ liệu cho Stack.
- Chức năng: Nhập dữ liệu cho các phần tử của stack (Dùng chèn đầu)
7. Xuất dữ liệu của Stack.
- Chức năng: Xuất dữ liệu của các phần tử stack ra màn hình

D. Luyện tập

Bài 1:

Tổ chức chương trình menu thực hiện trên STACK, dữ liệu của mỗi node là số nguyên với các chức năng :

1. Thêm một node ở đầu stack
2. Hủy một node ở đầu stack
3. Đếm số lần xuất hiện x trong stack.
4. Xóa toàn bộ những node có dữ liệu x trong stack.
5. Xem dữ liệu.

Dữ liệu của stack được cho trong tập tin “test.txt”:

7 0 4 3 8 6 3 1 8
(thực hiện như DSLK đơn)

Bài 2 :

Viết chương trình chuyển đổi số nguyên dương n hệ 10 sang hệ b (nguyên dương).

Input: n, // số hệ thập phân
b, //cơ số cần chuyển

Output: n cơ số b

Ví dụ:

Input: 16, // số hệ thập phân
2, //cơ số cần chuyển

Output: 10000 // (10000)₂

Mô tả thuật giải:

- B1: Khởi tạo STACK S rỗng
- B2: Lặp lại các bước sau cho đến khi số n bằng 0
 - Đẩy vào stack S phần dư của phép chia n cho b
 - Thương của n chia nguyên b đem gán cho n

B3: Xuất giá trị.

- Trong khi S chưa rỗng
Pop S và xuất giá trị vừa lấy được ra màn hình

Thực hiện:

- Tạo project Bai2D_Stack (chứa trong **MaSV_HoVaTen_Lab06**), với các tập **thuvien.h**, **Program.cpp**. Soạn thảo từng phần cho các tập tin trên.

- Nội dung tập tin “**thuvien.h**” :

```
//=====
typedef int data; //Kieu thanh phan du sieu cua nut
struct tagNode //Kieu Nut
{
    data        info;
    tagNode*    pNext;
};
typedef tagNode NODE;
struct STACK
{
    NODE* pHead; //Chua dc nut dau
    NODE* pTail; //Chua dc nut cuoi
};

//=====
NODE* GetNode(data x);
void CreatStack(STACK &s);
void Push(STACK &s, data x);
void XuatStack(STACK s);
void DoiCoSo(int n, int b);
//=====
//Ham tao nut co du lieu x, co next tro toi NULL
NODE* GetNode(data x)
{
    NODE *p;
    p = new NODE;
    if (p != NULL)
    {
        p->info = x;
        p->pNext = NULL;
    }
    return p;
}
//Tao DS rong
void CreatStack(STACK &s)
{
    s.pHead = s.pTail = NULL;
}
//Ham chen x vao dau s
void Push(STACK &s, data x)
{
    NODE* new_ele = GetNode(x);
    if (new_ele == NULL)
    {
        cout << "\nsoi cap phat bo nho! khong thuc hien duoc thao tac nay";
        return;
    }
    if (s.pHead == NULL)
    {
        s.pHead = new_ele; s.pTail = s.pHead;
    }
}
```

```

    }
    else
    {
        new_ele->pNext = s.pHead;
        s.pHead = new_ele;
    }
}
void XuatStack(STACK s)
{
    NODE *p;
    if (s.pHead == NULL)
    {
        cout << "\nStack rong!\n";
        return;
    }
    p = s.pHead;
    while (p != NULL)
    {
        cout << p->info << "t";
        p = p->pNext;
    }
}
//Ham doi co so : (n)10 sang (n)b
void DoiCoSo(int n, int b)
{
    int du;
    STACK s;
    CreatStack(s);
    cout << endl << n;
    while (n)
    {
        du = n % b;
        Push(s, du);
        n = n / b;
    }
    cout << " biieu dien dang co so (" << b << ") :\n";
    XuatStack(s);
}

```

Bài 3 :

Viết chương trình đảo ngược một xâu ký tự

Input : X;

Output Y;//dao nguoc X

Mô tả :

- Tạo Stack S rỗng.
- Trong khi chưa hết X:
 - Chèn ký tự vào đỉnh S;
 - Chuyển sang ký tự kế tiếp
- //Chuyển S vào Y
- Khởi tạo Y rỗng
- Trong khi chưa hết S:
 - Chèn ký tự của nút đang xét vào cuối Y;
 - Chuyển sang nút kế tiếp.

Thực hiện:

- Tạo project Bai3D_Stack (chứa trong **MaSV_HoVaTen_Lab06**), với các tập **thuvien.h**, **Program.cpp**. Soạn thảo từng phần cho các tập tin trên.

- Nội dung tập tin “**thuvien.h**” :

```
//=====
typedef char data; //Kieu thanh phan du sieu cua nut
struct tagNode //Kieu Nut
{
    data          info;
    tagNode*      pNext;
};
typedef tagNode NODE;
struct STACK
{
    NODE* pHead; //Chua dc nut dau
    NODE* pTail; //Chua dc nut cuoi
};
//=====
NODE* GetNode(data x);
void CreatStack(STACK &s);
void Push(STACK &s, data x);
void Chep_Chuoai_Sang_Stack(STACK &s, char X[MAX]);
void Chep_Stack_Sang_Chuoai(STACK s, char Y[MAX]);
//=====
void Chep_Chuoai_Sang_Stack(STACK &s, char X[MAX])
{
    CreatStack(s);
    if (X[0] == NULL)
        return;
    int i;
    for (i = 0; X[i] != NULL; i++)
        Push(s, X[i]);
}

void Chep_Stack_Sang_Chuoai(STACK s, char Y[MAX])
{
    Y[0] = NULL;
    if (s.pHead == NULL)
        return;
    NODE *p;
    p = s.pHead;
    int i = 0;
    while (p != NULL)
    {
        Y[i++] = p->info;
        p = p->pNext;
    }
    Y[i] = NULL;
}
```

Bài 4:

1. Cài đặt Thuật toán RPN (**Reverse Polish notation**) : Ký pháp nghịch đảo Ba Lan
2. Chuyển biểu thức đại số dạng trung tố (Infix) sang dạng hậu tố (Postfix)
(Xét trường hợp đơn giản : Toán hạng chỉ là các ký số, tức là các số từ 0 đến 9.)

Ví dụ :

- Dạng trung tố : $2*3+(6-4)$

- Dạng hậu tố : $2\ 3\ *\ 6\ 4\ -\ +$

- **Input :** sin (biểu thức trung tố, là một chuỗi . . .)
- **Output :** sout (biểu thức hậu tố, là một chuỗi . . .)
- **Mô tả thuật toán :**

B1. Khởi tạo stack s rỗng (dùng để chứa các toán tử);

B2. Lặp lại các việc sau cho đến khi dấu kết thúc biểu thức được đọc (NULL):

Đọc phần tử tiếp theo (hằng, biến, toán tử, ‘(’, ‘)’) trong biểu thức trung tố. Nếu phần tử là:

- Toán hạng (hằng hoặc biến): Cho ra output.
- Dấu ‘(‘: đẩy nó vào S;
- Dấu ‘)’ : lấy các toán tử trong stack ra và cho vào output cho đến khi gặp dấu ngoặc mở “(“.
(Dấu ngoặc mở cũng phải đưa ra khỏi stack nhưng không cho vào Output)
- Toán tử:
 - Trong khi đỉnh stack còn là toán tử và toán tử đó có độ ưu tiên **lớn hơn hoặc bằng** toán tử hiện tại thì lấy toán tử đó ra khỏi stack và cho vào output
 - Đưa toán tử hiện tại vào stack.

B3. Khi đạt đến dấu kết thúc biểu thức thì lấy ra và hiển thị các toán tử của S cho đến khi S rỗng;

+ Lưu ý về độ ưu tiên của các toán tử số học 2 ngôi :

Các toán tử 2 ngôi : ‘*’, ‘/’, ‘%’ có cùng độ ưu tiên và có độ ưu tiên cao hơn các toán tử sau (cùng độ ưu tiên)
‘+’, ‘-’.

Qua mô tả thuật toán, ta cần phải thực hiện :

- Định nghĩa kiểu dữ liệu STACK, với thành phần dữ liệu là ký tự.
- Cài đặt các thao tác cơ bản trên STACK : Tạo nút mới, tạo stack rỗng, chèn đầu, hủy đầu.
- Kiểm tra ký tự có phải là ký số (toán hạng)
- Kiểm tra ký tự có phải là toán tử.
- Xác định độ ưu tiên các toán tử

Thực hiện:

- Tạo project Bai4D_Stack (chứa trong **MaSV_HoVaTen_Lab06**), với các tập **thuvien.h**, **Program.cpp**. Soạn thảo từng phần cho các tập tin trên.

- Gợi ý nội dung tập tin **thuvien.h**:

+ Định nghĩa kiểu STACK như bài 3 (thành phần data của các nút là char)

+ Các hàm sau như bài 3:

```
NODE* GetNode(data x);  
void CreatStack(STACK &s);  
void Push(STACK &s, data x);
```

//Hàm hủy nút ở đỉnh stack

```
data Pop(STACK &s)  
{  
    data x;  
    NODE *p;  
    if (s.pHead == NULL)  
    {  
        return NULLDATA;  
    }  
    x = s.pHead->info;  
    p = s.pHead;  
    s.pHead = s.pHead->pNext;  
    delete p;  
    if (s.pHead == NULL)  
        s.pTail = NULL;  
    return x;  
}  
//=====
```

Input c //Ký tự

Output : uutien;

Mô tả :

Nếu (c=='*' || x=='/' || c=='%') thì uutien = 2;

Ngược lại, nếu (c=='+' || c=='-') thì uutien = 1;

Ngược lại uutien = 0;

Nguyên mẫu : int Do_UuTien_ToanTu(char c)


```
// ham kiem tra toan tu
Input c //Ky tu
Output : kq;
Mô tả : Nếu c là toán tử thì kq = 1
        Ngược lại kq = 0;
Nguyên mẫu : int LaToanTu(char c)

//Hàm chèn ký tự c vào cuối chuỗi a
Input : a, // Chuỗi
        c; ký tự
Output: a
Nguyên mẫu : void Chen_Cuoi_Chulai(char a[MAX], char c);

// Ham Kiem tra toan hang - chỉ cần kiểm tra có phải là ký số
Input : c; ký tự
Output: kq = 1 nếu c là ký số;
        Kq = 0; ngược lại;
Nguyên mẫu : int LaKySo(char c);

// ham chuyen tu trung to sang hau to
void TrungTo_HauTo(char sin[MAX], char sout[MAX])
{
    STACK s; //stack lưu các toán tử
    char c; //Ky tu hiện hành - đang xét
    data x; //lưu toán tử ở đỉnh stack trong thao tác Pop(s);
    int i; //duyet chuoi vao : sin
    CreatStack(s);
    // duyet tu trai sang phai cac ptu cua bieu thuc trung to P
    for (i = 0; sin[i] != NULL; i++)
    {
        c = sin[i]; //ky tu đang xét
        if (LaKySo(c) == 1) //la ky so : toan hang
            Chen_Cuoi_Chulai(sout, c); //chen c vao cuoi sout
        else
            if (c == '(')
                Push(s, c); //day '(' vao stack s
            else
                if (c == ')')
                {
                    x = Pop(s);
                    while (x != '(')
                    {
                        Chen_Cuoi_Chulai(sout, x);
                        x = Pop(s);
                    }
                }
            else // la toan hang
            {
                while (s.pHead != NULL && LaToanTu(s.pHead->info) == 1)
                    if (Do_UuTien_ToanTu(s.pHead->info) >= Do_UuTien_ToanTu(c))
                    {
                        x = Pop(s);
                        Chen_Cuoi_Chulai(sout, x);
                    }
                else
                    Break;
                //Push toán tử đang xét vào s
                Push(s, c);
            }
        }
    }
```

```

    }
    //da het sin
    while (s.pHead != NULL) //lay cac toan tu trong s chen vao cuoi sout
    {
        x = Pop(s);
        Chen_Cuoi_Chanoi(sout, x);
    }
}

```

- Tập tin **“Prpgram.cpp”**

...

Bài 5:

Cài đặt *Thuật toán đánh giá biểu thức đại số dạng RPN*

Mô tả thuật toán:

Input : Biểu thức hậu tố

Output: Giá trị biểu thức

Mô tả :

- B1. Khởi tạo STACK *s* rỗng;
- B2. Lặp lại các việc sau cho đến khi dấu kết thúc biểu thức được đọc:
 - . Đọc ký tự (toán hạng, toán tử) tiếp theo trong biểu thức;
 - + Nếu ký tự là toán hạng: đẩy nó vào *s*;
 - + Ngược lại: // ký tự là toán tử
 - Lấy từ đỉnh *s* hai toán hạng;
 - Áp dụng toán tử đó vào 2 toán hạng (theo thứ tự ngược);
 - Đẩy kết quả vừa tính trở lại *S*;
- B3. Khi gặp dấu kết thúc biểu thức, giá trị của biểu thức chính là giá trị ở đỉnh *S*.

Thực hiện:

- Tạo project Bai5D_Stack (chứa trong **MaSV_HoVaTen_Lab06**), với các tập **thuvien.h**, **Program.cpp**. Soạn thảo từng phần cho các tập tin trên.

- Nội dung tập tin **“thuvien.h”** :

```

//=====
- Gợi ý nội dung tập tin thuvien.h:
+ Định nghĩa kiểu STACK như bài 2 (thành phần data của các nút là int)
+ Các hàm sau như bài 2:
    NODE* GetNode(data x);
    void CreatStack(STACK &s);
    void Push(STACK &s, data x);
+ Hàm sau như bài 3 (nhưng thành phần data của các nút là int)
    data Pop(STACK &s);
+ Các hàm sau như bài 4:
// ham kiem tra toan tu
    int LaToanTu(char c);
//Kiem tra toan hang - chi can kiem tra co phai la ky so
    int LaKySo(char c);
+ các hàm khác :
//Đổi ký tự số thành số
    int So(char c)
    {
        return c - '0';
    }
//Tính giá trị biểu thức đại số dạng hậu tố
    int Tinh_BT_HauTo(char a[MAX])
    {
        int i;
        char c;

```

```
data x,y;
STACK s;
CreatStack(s);
for (i = 0; a[i] != NULL; i++)
{
    c = a[i];
    if (LaKySo(c) == 1)
    {
        x = So(c);
        Push(s, x);
    }
    else //toan tu
    {
        x = Pop(s);
        y = Pop(s);
        switch (c)
        {
            case '+':
                Push(s, y + x);
                break;
            case '-':
                Push(s, y - x);
                break;
            case '*':
                Push(s, y * x);
                break;
            case '/':
                Push(s, y / x);
                break;
            case '%':
                Push(s, y % x);
                break;
        }
    }
} //Ket thuc bieu thuc hau to
return s.pHead->info; //Gia tri bieu thuc hau to
}
```

- Tập tin **"Prpgram.cpp"**

E. Bài tập

Bài 1:

Viết chương trình tùy chọn thực hiện các phép toán trên các số nguyên lớn :

1. Tổng 2 số nguyên lớn
2. Hiệu 2 số nguyên
3. Tích 2 số nguyên
4. Xem các số
5. Chọn các số khác

Các số nguyên a, b cho trong các tập tin sau :

- Tập tin **"a.txt"**:
12213435408976545098909807650000
- Tập tin **"b.txt"** :
13435400

Bài 2:

1. Như bài 4.D, nhưng mở rộng 1 chút : :
2. Toán hạng có thể là các ký số, có thể là các ký tự từ 'A' đến 'Z', từ 'a' đến 'z'.

Bài 3 :

Viết chương trình nhập vào một biểu thức đại số (chuỗi) rồi tính giá trị của biểu thức. Sử dụng ký pháp nghịch đảo Ba Lan. Chỉ xét trường hợp đơn giản : Toán hạng chỉ là các ký số, tức là các số từ 0 đến 9.)

(Kết hợp bài 4, 5 phần D)

LAB 06. Danh sách liên kết đơn : Hàng đợi (Queue)

A. Mục tiêu

- Thực hiện các thao tác cơ bản trên QUEUE tổ chức bằng danh sách liên kết đơn.
- Cài đặt một số ứng dụng trên QUEUE.

B. Yêu cầu

- Phần ôn tập (C) : sinh viên tự ôn.
- **Buổi thực hành 7** (1 tiết cuối / 3 t)
 - Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : Bài 2,3,4 D (luyện tập)
 - Yêu cầu lưu trữ
 - ✓ Trong thư mục **MaSV_HoVaTen_Lab06** , tạo các project cho mỗi bài
 - ✓ Xóa thư mục debug trong project
 - ✓ Nén thư mục
 - Hình thức thu bài
 - ✓ Hệ thống thu bài trong phòng lab

C. Ôn tập

I. Cách tổ chức kiểu dữ liệu QUEUE bằng DSLK đơn

<ul style="list-style-type: none">• QUEUE (Hàng đợi) là kiểu dữ liệu tuyến tính nhằm biểu diễn các đối tượng được xử lý theo kiểu "vào trước ra trước" (FIFO: First In, First Out).• Dùng danh sách liên kết đơn để biểu diễn QUEUE• Các phép toán thêm vào và lấy ra được thực hiện cùng ở hai đầu danh sách : thêm vào cuối danh sách, lấy ra ở đầu danh sách.	<ul style="list-style-type: none">• Kiểu của thành phần dữ liệu của các nút trong QUEUE (giả sử là kiểu int): <code>typedef int data;</code>• Kiểu Phần tử (Nút) của tập hợp (DSLK): - Định nghĩa ban đầu <pre>struct tagNode { data info; tagNode* pNext; };</pre> - Đổi lại tên: <code>typedef tagNode NODE;</code>• Kiểu tập hợp QUEUE: <pre>struct QUEUE { NODE* pHead; NODE* pTail; };</pre>
--	--

II. Các thao tác cơ bản trên QUEUE

1. Tạo nút mới:
 - Chức năng: Tạo ra một nút của Queue với thành phần dữ liệu là x (Thành phần liên kết của nút mới là con trỏ có giá trị NULL)
 - Input : x
 - Output : trả về con trỏ - lưu trữ địa chỉ của pt vừa tạo (nếu tạo thành công) - có giá trị NULL (nếu ngược lại).
 - Nguyên mẫu của hàm: **NODE* GetNode(data x);**
2. Khởi tạo QUEUE rỗng:
 - Chức năng: Tạo ra một QUEUE s rỗng.

- Input : s
 - Output :s
 - Nguyên mẫu của hàm: **void CreatQUEUE(Queue &q);**
3. Thêm một giá trị vào cuối QUEUE (cuối DSLK đơn)
 - Chức năng: Tạo nút với dữ liệu x, chèn nút này vào cuối QUEUE,
 - Input : x , q
 - Output : q
 - Nguyên mẫu của hàm: **void EnQueue (Queue q, data x);**
 4. Lấy giá trị ở đầu QUEUE (Hủy nút đầu DSLK đơn)
 - Chức năng: Hủy nút ở đầu queue (trả về dữ liệu của đầu QUEUE)
 - Input : q
 - Output :x
 - Nguyên mẫu của hàm: **data DeQueue (Queue &q);**
 5. Xem thông tin ở đỉnh QUEUE (xuất dữ liệu nút đầu, không hủy)
 - Chức năng: Xem thông tin ở đỉnh QUEUE
 - Input : s
 - Output : x //dữ liệu đỉnh QUEUE
 - Nguyên mẫu của hàm: **data Top(Queue s);**
 6. Nhập dữ liệu cho QUEUE (từ tập tin hay từ bàn phím)
 - Chức năng: Nhập dữ liệu cho các phần tử của QUEUE (Dừng chèn cuối)
 7. Xuất dữ liệu của QUEUE.
 - Chức năng: Xuất dữ liệu của các phần tử QUEUE ra màn hình

D. Luyện tập

Bài 1:

Tổ chức chương trình menu thực hiện trên QUEUE, dữ liệu của mỗi node là số nguyên với các chức năng :

6. Thêm một giá trị vào cuối QUEUE
7. Hủy node đầu QUEUE
8. Đếm số lần xuất hiện x trong QUEUE.
9. Xóa toàn bộ những node có dữ liệu x trong QUEUE.
10. Xem dữ liệu.

Dữ liệu của QUEUE được cho trong tập tin “test.txt”:

7 0 4 3 8 6 3 1 8

(thực hiện như DSLK đơn)

Bài 2 :

Cho một chuỗi ký tự, trong đó chỉ gồm có chữ cái HOA và dấu *.

- Với chữ cái : tượng trưng cho thao tác thêm chữ cái tương ứng vào cuối một queue.
- Với dấu * : tượng trưng cho thao tác lấy nội dung phần tử ở đầu queue và xuất ra màn hình.

Thực hiện chuỗi thao tác trên.

Viết chương trình cho biết kết quả của queue sau mỗi thao tác.;

- Chuỗi ký tự các thao tác cho trong tập tin “test.txt” :

EAS*Y**QUE***ST***I*ON

Thực hiện:

- Tạo project Bai2D_Queue (chứa trong **MaSV_HoVaTen_Lab06**), với các tập **thuvien.h**, **Program.cpp**. Soạn thảo từng phần cho các tập tin trên.

- Nội dung gợi ý tập tin “**thuvien.h**” :

//=====

```

#define MAX 100
#define NULLDATA ''
typedef char data; //Kieu thanh phan du lieu cua nut
struct tagNode //Kieu Nut
{
    data          info;
    tagNode*      pNext;
};
typedef tagNode NODE;
//Kieu DSLK don
struct QUEUE
{
    NODE* pHead; //Chua dc nut dau
    NODE* pTail; //Chua dc nut cuoi
};
//=====
NODE* GetNode(data x);
void CreatQueue(QUEUE &q);
void EnQueue(QUEUE &q, data x);
data DeQueue(QUEUE &q)
{
    NODE *p;
    data x;
    if (q.pHead == NULL)
        return NULLDATA;
    p = q.pHead;
    x = p->info;
    q.pHead = q.pHead->pNext;
    delete p;
    if (q.pHead == NULL)
        q.pTail = NULL;
    return x;
}
void File_String(char *f, char a[MAX]); //Tap tin -> Chuoi
void XuatQueue(QUEUE q);

void XuLy_ThaoTac(char a[MAX])
{
    QUEUE q;
    CreatQueue(q);
    char c;
    int i;
    for (i = 0; a[i] != NULL; i++)
    {
        c = a[i];
        if ('A' <= c && c <= 'Z')
        {
            EnQueue(q, c);
            XuatQueue(q);
        }
        else
            if (c == '*')
            {
                if (DeQueue(q));
                XuatQueue(q);
            }
    }
    cout << "\nKet qua sau khi thuc hien chuoi thao tac \"" << a << "\":\n";
    XuatQueue(q);
}

```

```
cout << endl;
system("PAUSE");
}
```

Bài 3:

Sắp tăng danh sách số nguyên bằng thuật toán Radix (các lô là các queue)

- Dữ liệu cho trong tập tin **"Test1.txt"** :

7013 8425 1239 428 1424 7009 4518 3252 9170 999 1725 701

- Dữ liệu cho trong tập tin **"Test2.txt"** :

7013 8425 1239 428 1424 7009 4518 3252 9170 999
 1725 701 17013 84251 12391 4280 14245 70093 45 453252
 9170 9909 17205 12370 7013 238425 1239 428 1424 7009
 18 325 29170 999 1725 701 32345 23126 123 34

Thực hiện:

- Tạo project Lab06_Bai3D_Queue (chứa trong **MaSV_HoVaTen_Lab06_Queue**) , với các tập **thuvien.h**, **Program.cpp**. Soạn thảo từng phần cho các tập tin trên.

- Nội dung gợi ý tập tin **"thuvien.h"** :

typedef int data; //Kieu thanh phan du lieu cua nut

struct tagNode //Kieu Nut

```
{
    data info;
    tagNode* pNext;
};
```

typedef tagNode NODE;

struct QUEUE

```
{
    NODE* pHead;
    NODE* pTail;
};
```

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

NODE* GetNode(data x);

void CreatQueue(QUEUE &q);

void EnQueue(QUEUE &q, data x);

data DeQueue(QUEUE &q);

void TapTin_Queue(char *f, QUEUE &q);

void XuatQueue(QUEUE q);

void Radix(QUEUE &q);

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

void Radix(QUEUE &q)

```
{
    int k, j, du, thuong;
    if (q.pHead == NULL)
    {
        cout << "\nDS rong";
        return;
    }
    int max, //Gia tri lon nhat trong q
        m; //so ky so lon nhat trong cac gia tri cua q
    max = q.pHead->info;
    m = 0;
    //Tim max(q)
    NODE *p;
    for (p = q.pHead->pNext; p != NULL; p = p->pNext)
        if (p->info > max)
            max = p->info;
    //Xac dinh so cac chu so cua max(a) : m
    while (max != 0)
```

```
{
    max = max / 10;
    m++;
}
//Khai báo 10 lo : queue
QUEUE q0, q1, q2, q3, q4, q5, q6, q7, q8, q9;
//Khởi tạo các queue rỗng
CreatQueue(q0);
CreatQueue(q1);
CreatQueue(q2);
CreatQueue(q3);
CreatQueue(q4);
CreatQueue(q5);
CreatQueue(q6);
CreatQueue(q7);
CreatQueue(q8);
CreatQueue(q9);

k = 0; //khởi tạo chu số k = 0 : hàng đơn vị
while (k < m) // dừng khi k = m
{
    while (q.pHead != NULL)
    {
        //xác định chu số hàng k của a[i] : du
        thuong = q.pHead->info;
        for (j = 0; j <= k; j++)
        {
            du = thuong % 10;
            thuong = thuong / 10;
        }
        //Phân vào các lo
        switch (du)
        {
            case 0:
                EnQueue(q0, q.pHead->info); //thêm vào cuối q0
                DeQueue(q); //hủy nút đầu q
                break;
            case 1:
                EnQueue(q1, q.pHead->info);
                DeQueue(q);
                break;
            case 2:
                EnQueue(q2, q.pHead->info);
                DeQueue(q);
                break;
            case 3:
                EnQueue(q3, q.pHead->info);
                DeQueue(q);
                break;
            case 4:
                EnQueue(q4, q.pHead->info);
                DeQueue(q);
                break;
            case 5:
                EnQueue(q5, q.pHead->info);
                DeQueue(q);
                break;
            case 6:
                EnQueue(q6, q.pHead->info);
```



```
        DeQueue(q);
        break;
    case 7:
        EnQueue(q7, q.pHead->info);
        DeQueue(q);
        break;
    case 8:
        EnQueue(q8, q.pHead->info);
        DeQueue(q);
        break;
    case 9:
        EnQueue(q9, q.pHead->info);
        DeQueue(q);
        break;
    }
} //Phan xong vào các lo; và có q rỗng

//Nối các queue lại theo trình tự
while (q0.pHead != NULL)
{
    EnQueue(q, q0.pHead->info); //thêm vào cuối q
    DeQueue(q0); //hủy nút đầu q0
} //q0 rỗng
while (q1.pHead != NULL)
{
    EnQueue(q, q1.pHead->info);
    DeQueue(q1);
}
while (q2.pHead != NULL)
{
    EnQueue(q, q2.pHead->info);
    DeQueue(q2);
}
while (q3.pHead != NULL)
{
    EnQueue(q, q3.pHead->info);
    DeQueue(q3);
}
while (q4.pHead != NULL)
{
    EnQueue(q, q4.pHead->info);
    DeQueue(q4);
}
while (q5.pHead != NULL)
{
    EnQueue(q, q5.pHead->info);
    DeQueue(q5);
}
while (q6.pHead != NULL)
{
    EnQueue(q, q6.pHead->info);
    DeQueue(q6);
}
while (q7.pHead != NULL)
{
    EnQueue(q, q7.pHead->info);
    DeQueue(q7);
}
while (q8.pHead != NULL)
```

```

    {
        EnQueue(q, q8.pHead->info);
        DeQueue(q8);
    }
    while (q9.pHead != NULL)
    {
        EnQueue(q, q9.pHead->info);
        DeQueue(q9);
    }
    //Đa cap nhap lai q theo thu tu tang dan hang k
    k++; //xet hang k lon hon ke tiep
}
}

```

Bài 4 : (Lập lịch xoay vòng).

Một lớp học có n nhóm, mỗi nhóm thực hiện trực lớp 1 ngày và thực hiện xoay vòng theo thứ tự cho sẵn trong m ngày.

Thứ tự trực của các nhóm, do chọn ngẫu nhiên, hoặc phân công, hoặc đăng ký trước thì được quyền chọn.

Thông tin các nhóm có thể nhập từ bàn phím, hay từ tập tin.

Viết chương trình minh họa việc lập lịch xoay vòng trên.

- Tập tin dữ liệu cho trong **“Test.txt”**, định dạng trong nhiều hàng (số hàng tương ứng cho số nhóm), mỗi hàng có 2 giá trị, giá trị đầu là chuỗi biểu diễn nhóm số, giá trị thứ 2 là thứ tự trực của nhóm :

```

=====
N01    5
N02    3
N03    7
N04    2
N05    1
N06    4
N07    6
=====

```

Chương trình tổ chức như sau :

- Chuyển tập tin “Test.txt” sang queue q.
- Sắp q tăng dần theo thứ tự trực (số nhỏ trực trước)
- **Lập (điều kiện dừng do số ngày):**

- **c = Dequeue(q);**
- **Xuất c;**
- **Enqueue(q,c);**

Thực hiện:

- Tạo project Lab06_Bai4D_Queue (chứa trong **MaSV_HoVaTen_Lab06_Queue**) , với các tập **thuvien.h**, **Program.cpp**. Soạn thảo từng phần cho các tập tin trên.

- Nội dung gợi ý tập tin **“thuvien.h”** :

```

struct nhom
{
    char ns[5];
    int stt;
};
typedef nhom data; //Kieu thanh phan du lieu cua nut
struct tagNode //Kieu Nut
{
    data info;
    tagNode* pNext;
};
typedef tagNode NODE;
//Kieu DSLK don
struct QUEUE
{
    NODE* pHead;

```

```

NODE* pTail;
};
//=====
NODE* GetNode(data x);
void CreatQueue(Queue &q);
void EnQueue(Queue &q, data x);
data DeQueue(Queue &q);
void TapTin_Queue(char *f, Queue &q);
int SoNhom(Queue q);
void XuatQueue(Queue q);
void Sap_Tang_TTTruc(Queue &q); //sap tang q theo thu tu truc
void XuatLichTruc(Queue &q, int m); //Xep va xuat Lich truc xoay vong n nhom trong m ngay
//=====
//sn nhom truc xoay vong trong m ngay
void XuatLichTruc(Queue &q, int m)
{
    int i,sn;
    data x;
    int vong= 1;
    sn = SoNhom(q); // so nhom : so nut trong q
    Sap_Tang_TTTruc(q); //dap tang q theo thu tu truc
    cout << "\nVong " << vong << " : \n";
    for (i = 1; i <= m; i++)
    {
        cout << setiosflags(ios::left);
        cout << setw(13)<< "\nTruc ngay thu " << setw(4) << i << setw(10)<< " la nhom : ";
        x = DeQueue(q);
        cout << setw(5);
        cout << x.ns;
        EnQueue(q, x);
        if (i % sn == 0)
        {
            cout << "\n===== ";
            vong++;
            cout << "\nVong " << vong << " : \n";
        }
    }
    cout << endl;
}
}

```

E. Bài tập

Bài 1:

Mô phỏng việc tạo buffer in một file ra máy in.

Buffer xem như là một queue, khi ra lệnh in file máy tính sẽ thực hiện một cách lặp quá trình sau cho đến hết file:

- Đưa nội dung của tập tin vào buffer cho đến khi buffer đầy hoặc hết file.
- In nội dung trong buffer ra máy in cho tới khi hàng rỗng.

Hãy mô phỏng quá trình trên để in một file văn bản lên từng trang màn hình.

Bài 2 :

Để quản lí thu chi ngoại tệ, người ta cần lưu các chứng từ, thông tin trong bảng chứng từ gồm:

- ngày /tháng/năm thu/chi ngoại tệ
- số lượng
- đơn vị tiền tệ
- tỷ giá
- loại nhập/xuất
- thành tiền VND

Viết chương trình xử lí thu /chi theo phương pháp FIFO

LAB 07. Danh sách liên kết đơn vòng

A. Mục tiêu

- Tìm hiểu cách tổ chức kiểu DSLK đơn vòng
- Thực hiện các thao tác cơ bản trên DSLK đơn vòng.
- Các ứng dụng liên quan

B. Yêu cầu

- Phần ôn tập (C) : sinh viên tự ôn.

- **Buổi thực hành 8** (3 t)

- 2 tiết đầu : Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : **Bài 1, 2 D (luyện tập)**
 - Yêu cầu lưu trữ
 - ✓ Tạo thư mục **MaSV_HoVaTen_Lab07_LLlist** chứa trong ổ đĩa qui định.
 - ✓ Mỗi bài, tạo project lưu trữ trong thư mục trên.
 - ✓ Xóa thư mục debug trong mỗi project.
 - ✓ Nén thư mục
 - Hình thức thu bài
 - ✓ Hệ thống thu bài trong phòng lab
- Tiết cuối : Kiểm tra giữa kỳ

C. Ôn tập

I. Tổ chức kiểu dữ liệu DSLK đơn vòng:

- DSLK đơn vòng tổ chức như DSLK đơn, chỉ khác một điểm là liên kết của con trỏ Tail chứa địa chỉ nút đầu (tức là : l.pTail -> pNext == l.pHead).
Ta sẽ đặt tên Kiểu DSLK đơn vòng (khi cài đặt) là : **LLIST**

II. Cài đặt kiểu dữ liệu DSLK đơn vòng:

- Kiểu của thành phần dữ liệu trong DS:
(Giả sử là kiểu int)
typedef int data;
- Kiểu Phần tử (Nút) của DS:
 - Định nghĩa ban đầu

```
struct tagNode
{
    data          info;
    tagNode*      pNext;
};
```
 - Đổi lại tên:
typedef tagNode NODE;

- Kiểu DSLK đơn vòng:

```
struct LLIST
{
    NODE* pHead; //Con trỏ lưu địa chỉ phần tử đầu DSLK
    NODE* pTail; //Con trỏ lưu địa chỉ phần tử cuối DSLK
};
```

III. Các thao tác thường dùng trên DSLK đơn vòng(trên các nút, trên danh sách)

(Giả sử ta đã cài đặt kiểu DSLK đơn vòng như trên)

1. Tạo nút mới: GetNode (x) :

- Chức năng: Tạo ra một phần tử (nút) của DSLK với thành phần dữ liệu là x
(Thành phần liên kết của nút mới là con trỏ có giá trị NULL)
- Input : x
- Output : trả về con trỏ - lưu trữ địa chỉ của pt vừa tạo (nếu tạo thành công) - có giá trị NULL (nếu ngược lại).
- Nguyên mẫu của hàm: **NODE* GetNode(data x);**

2. Khởi tạo DS rỗng: **CreatLlist (l)**

- Chức năng: Tạo ra một DSLK vòng l rỗng.
- Input : l
- Output : l
- Nguyên mẫu của hàm: void CreatLlist(LLIST &l);

3. Duyệt danh sách:

- Chức năng: Duyệt danh sách từ đầu DS để xử lý dữ liệu của nút (có thể là xuất dữ liệu ra màn hình, đếm số nút, . . .)
- Input : l
- Nguyên mẫu của hàm: void ProcessLlist (LLIST l);

4. Tìm nút có info là x:

- Chức năng: Tìm trong DS vòng có nút chứa thành phần Info là x?
- Input : l, x
- Output : p (con trỏ p chứa nút đầu tiên có info là x); nếu có NULL; ngược lại
- Nguyên mẫu của hàm: NODE *Search(LLIST l, data x);

5. Chèn nút (đã có) vào đầu DSLK đơn vòng:

- Chức năng: Chèn một nút (đã có trước) vào đầu DS (Chú ý: thành phần liên kết của nút này là con trỏ có giá trị NULL)
- Input : l, new_ele
- Output : l (Nút đầu là new_ele)
- Nguyên mẫu của hàm: void AddHead(LLIST &l, NODE* new_ele);

6. Chèn một giá trị dữ liệu vào đầu DSLK đơn vòng:

- Chức năng: Tạo trước nút new_ele có info là x, con trỏ liên kết có giá trị NULL; sau đó Chèn nút này vào đầu DS.
- Input : l, x (có kiểu data)
- Output : - l (Nút đầu có info là x kiểu data)
- Nguyên mẫu của hàm: void InsertHead(LLIST &l, data x);

7. Chèn nút (đã có) vào Cuối DSLK đơn vòng:

- Chức năng: Chèn một nút (đã có trước) vào cuối DS (Chú ý: thành phần liên kết của nút này là con trỏ có giá trị NULL)
- Input : l, new_ele
- Output : l (Nút cuối là new_ele)
- Nguyên mẫu của hàm: void AddTail(LLIST &l, NODE *new_ele);

8. Chèn một giá trị dữ liệu vào cuối DSLK đơn vòng:

- Chức năng: Tạo nút new_ele có info là x, con trỏ liên kết có giá trị NULL; sau đó chèn nút này vào cuối DS.
- Input : l, x (có kiểu data)
- Output : - l (Nút cuối có info là x kiểu data)
- Nguyên mẫu của hàm: void InsertTail (LLIST &l, data x);

9. Chèn một nút (chưa có trước) vào sau nút do con trỏ q trỏ tới :

- Chức năng: Tạo nút new_ele có info là x, con trỏ liên kết có giá trị NULL; sau đó chèn nút này vào sau nút do con trỏ q trỏ tới.
- Input : l, q
- Output : l (Nút cuối có info là x kiểu data)
- Nguyên mẫu của hàm: void InsertAfter(LLIST &l, NODE *q, data x);

10. Hủy nút đầu ra khỏi DSLK đơn vòng:

- Chức năng: Hủy nút đầu ra khỏi DSLK đơn vòng
- Input : l

- Output : 1 (bớt nút đầu của 1 input)
 - Nguyên mẫu của hàm: void RemoveHead(LLIST &l);
11. Hủy nút ở vị trí sau nút do con trỏ q trở tới :
- Chức năng: Hủy nút sau nút có vị trí do con trỏ q trở tới.
 - Input : l, q
 - Output : 1 (bớt 1 nút)
 - Nguyên mẫu của hàm: void RemoveAfter (LLIST &l, NODE *q);
12. Hủy nút có thành phần info là x :
- Chức năng: Hủy nút có info là x.
 - Input : l, x
 - Output : 1; Nếu có nút
0; Nếu ngược lại
 - Nguyên mẫu của hàm: ~~int RemoveNode(LLIST &l, data x);~~
0
13. Hủy toàn bộ danh sách vòng
- Chức năng: Hủy toàn bộ danh sách
 - Input : l
 - Output : l rỗng
 - Nguyên mẫu của hàm: void RemoveList(LLIST &l);

D. LUYỆN TẬP

Bài 1:

Viết chương trình tùy chọn thực hiện các thao tác cơ bản trên danh sách liên kết đơn vòng, dữ liệu của các nút trong danh sách là số nguyên:

```
//Tìm kiếm
1. Tạo dữ liệu
1. Xem dữ liệu
1. Tìm x - Có/Không
2. Tìm x - Tra về vị trí nút đầu tiên xuất hiện nếu có
3. Tìm x - Tra về vị trí nút cuối cùng xuất hiện nếu có
4. Tìm x - Xuất các vị trí xuất hiện nếu có";
5. Xuất DS mới, với Head mới trở tới nút x xuất hiện cuối cùng";
//Chen
6. Chen x vào đầu danh sách vòng
7. Chen x vào cuối danh sách vòng
//Huy
8. Huy nút đầu danh sách vòng
9. Huy nút cuối danh sách vòng
10. Huy toàn bộ danh sách vòng";
11. Dem so nut danh sach vong";
```

Dữ liệu được cho trong các tập tin :

- "Test1.txt" :

9 7 1 7 -1 4 7 -4 8 1 7 1 7 8
0

- "Test2.txt"

0 2 9 7 6 3 6 7 6 0

Thực hiện:

- Tạo project Bai1D (chứa trong **MaSV_HoVaTen_Lab07_Llist**), với các tập **Header.h,thuvien.h, Program.cpp**.Soạn thảo từng phần cho các tập tin trên.

Trong đó:

- Tập tin header.h : Chứa các hằng, định nghĩa kiểu dữ liệu DSLKDV LLIST, các thao tác cơ bản trên DSLK đơn vòng.
- Tập tin thuvien.h : Chứa các chức năng của bài toán yêu cầu
- Tập tin menu.h : Chứa các hàm tổ chức xử lý menu
- Tập tin Program.cpp : Tập tin chương trình, điều khiển việc lặp thực hiện các chức năng chương trình trong hệ thống menu.

- Nội dung tập tin “**Program.cpp**” :

```
//Cac thu vien
#include <iostream>
#include <fstream>
#include <conio.h>
#include <math.h>

using namespace std;
#include "Header.h"
#include "ThuVien.h"
#include "Menu.h"

void ChayChuongTrinh();
//=====
void ChayChuongTrinh()
{
    int soMenu = MAX_MENU,
        menu;
    LLIST l;
    char filename[MAX];
    //Dieu khien viec chon file cho dung
    do
    {
        system("CLS");
        cout << "\nNhap ten tap tin, filename = ";
        _flushall();
        cin >> filename;
        if (!File_LLIST(filename, l))
        {
            cout << "\nLoi mo file ! nhap lai\n";
            _getch();
        }
        else
        {
            cout << "\nDu lieu tap tin " << filename << " da duoc chuyen vao DSLKDV l"
                << "\nNhan phim bat ky de tiep tuc";
            _getch();
            break;
        }
    } while (1);

    //Dieu khien menu
    do
    {
        system("CLS");
        menu = ChonMenu(soMenu);
        XuLyMenu(menu, l);
        system("PAUSE");
    } while (menu > 0);
}
```

```
int main()
{
    ChayChuongTrinh();
    return 1;
}
```

- Nội dung tập tin “header.h” :

```
//Header DSLK vòng : Định nghĩa và các thao tác cơ bản
#define MAX_MENU 13
#define MAX 13
//Định nghĩa kiểu dữ liệu : DSLK đơn vòng
typedef int data; //Kiểu thành phần dữ liệu của nút
struct tagNode //Kiểu các Nút của danh sách LK đơn vòng
{
    data info;
    tagNode* pNext;
};
typedef tagNode NODE; //Đổi lại tên kiểu Nút
struct LLIST //Kiểu DSLK đơn
{
    NODE* pHead; //Chưa dc nút đầu
    NODE* pTail; //Chưa dc nút cuối
};

//=====
//Khai báo các hàm cơ bản
//=====

NODE* GetNode(data x);
void CreatLLIST(LLIST &l);
void InsertHead(LLIST &l, data x);
void InsertTail(LLIST &l, data x);

void Output_Llist(LLIST l);
int File_LLIST(char *f, LLIST &l);

//=====
//Định nghĩa các hàm cơ bản
//=====
//Hàm tạo nút có info là x, next có giá trị NULL
NODE* GetNode(data x)
{
    NODE *p;
    p = new NODE;
    if (p != NULL)
    {
        p->info = x;
        p->pNext = NULL;
    }
    return p;
}

//Tạo DS vòng
void CreatLLIST(LLIST &l)
{
    l.pHead = l.pTail = NULL;
}
```



```
//=====
//Chen
//Ham chen x vao dau l : Tao nut co info la x, next tro toi NULL, chen nut nay vao dau l
void InsertHead(LLIST &l, data x)
{
    NODE* new_ele = GetNode(x);
    if (new_ele == NULL)
    {
        cout << "\nLoi cap phat bo nho! khong thuc hien duoc thao tac nay";
        return;
    }
    if (l.pHead == NULL)//Chen vao DS rong
    {
        l.pHead = new_ele;
        l.pTail = l.pHead;
        l.pTail->pNext = l.pHead;
    }
    else
    {
        new_ele->pNext = l.pHead;
        l.pTail->pNext = new_ele;
        l.pHead = new_ele;
    }
}

//Chen cuoi
void InsertTail(LLIST &l, data x)
{
    NODE* new_ele = GetNode(x);
    if (new_ele == NULL)
    {
        cout << "\nLoi cap phat bo nho! khong thuc hien duoc thao tac nay";
        return;
    }
    if (l.pHead == NULL)//Chen vao DS rong
    {
        l.pHead = new_ele;
        l.pTail = l.pHead;
        l.pTail->pNext = l.pHead;
    }
    else
    {
        new_ele->pNext = l.pHead;
        l.pTail->pNext = new_ele;
        l.pTail = new_ele;
    }
}

//Nhap, xuat, he thong
//=====
void Output_Llist(LLIST l)
{
    NODE *p;
    if (l.pHead == NULL)
    {
        cout << "\nDS don vong rong!\n";
        return;
    }
}
```

```
    }
    p = l.pHead;
    do
    {
        cout << p->info << "t";
        p = p->pNext;
    } while (p != l.pHead); //Chua giap vong
}
```

//Tap tin -> LLIST

```
int File_LLIST(char *f, LLIST &l)
{
    ifstream in(f);    //Mo de doc
    if (!in)
        return 0;

    CreatLLIST(l);
    data x;
    in >> x;
    InsertTail(l, x);
    while (!in.eof())
    {
        in >> x;
        InsertTail(l, x);
    }
    in.close();
    return 1;
}
```

- Nội dung tập tin “**thuvien.h**” :

```
//Tim kiem co (1)/khong(0)
int Search(LLIST l, data x)
{
    NODE *p;
    p = l.pHead;
    do
    {
        if (p->info == x)
            return 1;
        p = p->pNext;
    } while (p != l.pHead); //Chua giap vong
    return 0;
}

=====
//Tim vi tri nut dau tien Pos co info la x; Nut dau tien cua DS dem tu 0
//Khong co: ham tra ve -1
//Co : Tra ve vi tri cua nut dau tien co info la x

int Search_Pos_First(LLIST l, data x)
{
    NODE *p;
    int kq = -1, //khong co
        i = 0;
    p = l.pHead;

    do
    {

```

```

        if (p->info == x)
        {
            kq = i;
            break;
        }
        p = p->pNext;
        i++;
    } while (p != l.pHead); //Chua giap vong
    return kq;
}

//=====
// Tim vi tri nut cuoi cung Pos co info la x; Nut dau tien cua DS dem tu 0
//Khong co: ham tra ve -1
//Co : Tra ve vi tri cua nut cuoi cung co info la x

int Search_Pos_End(LLIST l, data x)
{
    NODE *p;
    int kq = -1, //khong co
        i = 0;
    p = l.pHead;
    do
    {
        if (p->info == x)
            kq = i;
        p = p->pNext;
        i++;
    } while (p != l.pHead); //Chua giap vong

    return kq;
}

//=====
//Tim nut co info la x:
//Khong co: Thong bao khong
//Neu co, ham xuất tất cả các vị trí x xuất hiện
//Nut dau tien dem tu 0
void Search_All_Pos_x(LLIST l, data x)
{
    NODE *p;
    int i = 0;
    int kq = Search(l, x);

    if (!kq)
    {
        cout << endl << x << " không có trong danh sách";
        return;
    }
    cout << endl << x << " xuất hiện trong danh sách tại các vị trí :\n";

    p = l.pHead;
    do
    {
        if (p->info == x)
            cout << i << '\t';
        p = p->pNext;
        i++;
    } while (p != l.pHead); //Chua giap vong
}

```

```

}

//=====

//Co : tra ve dia chi nut cuoi cung co x
//Khong : NULL
NODE *Search_End(LLIST l, data x)
{
    NODE *p, *q = NULL;
    p = l.pHead;
    if (p == NULL);
    else
    {
        do
        {
            if (p->info == x)
                q = p;
            p = p->pNext;
        } while (p != l.pHead); //Chua giap vong
    }
    return q;
}

//=====
//DS moi co Head tro toi nut cuoi x xuất hiện
int Head_New(LLIST &l, data x)
{
    NODE *q, *r = NULL;
    if (l.pHead == NULL)
        return 0;
    q = Search_End(l, x);
    if (q == NULL)
        return -1;
    l.pHead = q;
    do
    {
        r = q;
        q = q->pNext;
    } while (q != l.pHead);

    l.pTail = r;
    l.pTail->pNext = l.pHead;
    return 1;
}

//=====
//Huy
//=====
//Huy nut dau
void RemoveHead(LLIST &l)
{
    NODE *p = l.pHead;
    if (p == NULL) return;
    if (l.pHead == l.pTail) // 1nut
        l.pHead = l.pTail = NULL;
    else
    {
        l.pHead = p->pNext; // Cap nhât lai Head
        l.pTail->pNext = l.pHead; //Cap nhât Next của Tail
    }
}

```

```

        delete p;
    }

//=====
//Huy nút cuối
void RemoveTail(LLIST &l)
{
    NODE *p, *q;
    if (l.pHead == NULL)
        return;
    //xác định địa chỉ nút đứng trước Tail : q
    p = l.pHead;
    q = NULL;
    do
    {
        q = p;
        p = p->pNext;
    } while (p != l.pTail);

    l.pTail = q; //q->pNext != NULL
    delete p;

    if (l.pTail == NULL) //Tail mới
        l.pHead = NULL;
    else
        l.pTail->pNext = l.pHead;
}

//=====
//Huy DS
void RemoveLLIST(LLIST &l)
{
    if (l.pHead == NULL)
        return;
    NODE *p;
    do
    {
        p = l.pHead;
        l.pHead = p->pNext; // Cập nhật lại Head
        l.pTail->pNext = l.pHead; //Cập nhật Next của Tail
        delete p;
    } while (l.pHead != l.pTail);
    p = l.pHead;
    l.pHead = l.pTail = NULL;
    delete p;
}

//=====
//Đếm số lượng nút
//int SoNut(LLIST l)
//=====
//Copy l sang l1
void Copy_Llist(LLIST l, LLIST &l1)
{
    CreatLLIST(l1);
    if (l.pHead == NULL)
        return;
    NODE *p;
    p = l.pHead;
    do

```

```
{
    InsertTail(l1, p->info);
    p = p->pNext;
} while (p != l.pHead);
}
```

Bài 2 (bài toán Josephus) :

Nhóm binh sĩ bị kẻ thù bao vây tổ chức chọn ra một binh sĩ đi cầu cứu.

Việc chọn thực hiện như sau: Đầu tiên chỉ ra một cách ngẫu nhiên một số nguyên dương n và một binh sĩ.

Các binh sĩ được sắp theo vòng tròn, và họ đếm bắt đầu từ binh sĩ được chọn ngẫu nhiên trước, khi đạt đến n , binh sĩ tương ứng này được lấy ra khỏi vòng và chuyển đếm lại bắt đầu từ binh sĩ tiếp theo.

Quá trình này cứ tiếp tục cho đến khi chỉ còn một binh sĩ. Đó là người sẽ được chọn để đi cầu cứu.

Hãy xuất các binh sĩ ra khỏi vòng theo thứ tự và binh sĩ được chọn đi cầu cứu.

Danh sách các binh sĩ (các chuỗi) cho trong tập tin **test.txt** :

```
001    002    003    004    005    006    007    008    009    010
011    012    013    014    015    016    017    108    019    020
021    022    023    024    025    026    027    028    029    030
031    032    033    034    035    036    037    038    039    040
041
```

Chương trình tổ chức theo thư viện và không có hệ thống menu, gồm các tập tin :

- **program.cpp** : điều khiển thực hiện chương trình (có thể lập thực hiện các tập tin khác nhau về binh sĩ)

- Tập tin **header.h** : Định nghĩa kiểu dữ liệu của thành phần nút (biểu diễn binh sĩ), định nghĩa kiểu dữ liệu danh sách liên kết đơn vòng (binh sĩ xếp vòng tròn), các thao tác cơ bản trên DSLK đơn vòng như : tạo nút mới, tạo DSLK đơn vòng rỗng, chèn dữ liệu (binh sĩ) vào cuối DSLK đơn vòng, chuyển tập tin (danh sách binh sĩ) vào DSLK đơn vòng, Xuất DS ra màn hình.

- Tập tin **thuvien.h** : Chứa các hàm xây dựng chương trình như : Tạo số ngẫu nhiên, hàm trả về con trỏ trỏ đến trước nút thứ n , hàm hủy nút sau nút q , đếm số nút, hàm chọn và xuất ra binh sĩ đi cầu cứu (và xuất ra các binh sĩ ra khỏi vòng)

- Các tập tin dữ liệu **test1.txt**, **test2.txt** : các chuỗi biểu diễn binh sĩ.

Gợi ý các tập tin trên như sau :

- Tập tin **header.h** :

```
//Header DSLK vòng : Định nghĩa và các thao tác cơ bản
#define MAX 30
//Định nghĩa kiểu dữ liệu thành phần của nút trong DSLK đơn vòng
typedef char data[MAX]; //Kiểu thành phần dữ liệu của nút
struct tagNode //Kiểu các Nút của danh sách LK đơn vòng
{
    data info;
    tagNode* pNext;
};
typedef tagNode NODE; //Đổi lại tên kiểu Nút
struct LLIST //Kiểu DSLK đơn
{
    NODE* pHead; //Chứa đc nút đầu
    NODE* pTail; //Chứa đc nút cuối
};
//=====
//Khai báo các hàm cơ bản
NODE* GetNode(data x);
void CreatLLIST(LLIST &l);
void InsertTail(LLIST &l, data x);

void Output_Llist(LLIST l);
int File_LLIST(char *f, LLIST &l);
//=====
//Định nghĩa các hàm cơ bản
```

//Ham tao nut co info la x, next co gia tri NULL

NODE* GetNode(data x)

```
{
    NODE *p;
    p = new NODE;
    if (p != NULL)
    {
        strcpy_s(p->info, x);
        p->pNext = NULL;
    }
    return p;
}
```

//Tao DS rong

void CreatLLIST(LLIST &l)

```
{
    l.pHead = l.pTail = NULL;
}
```

//=====

//Chen

//=====

//Ham chen x vao cuoi l : Tao nut co info la x, next tro toi NULL, chen nut nay vao cuoi l

void InsertTail(LLIST &l, data x)

```
{
    NODE* new_ele = GetNode(x);

    if (new_ele == NULL)
    {
        cout << "\nLoi cap phat bo nho! khong thuc hien duoc thao tac nay";
        return;
    }

    if (l.pHead == NULL) //Chen vao DS rong
    {
        l.pHead = new_ele;
        l.pTail = l.pHead;
        l.pTail->pNext = l.pHead;
    }
    else
    {
        new_ele->pNext = l.pHead;
        l.pTail->pNext = new_ele;
        l.pTail = new_ele;
    }
}
```

//Nhap, xuat

//=====

void Output_Llist(LLIST l)

```
{
    NODE *p;
    if (l.pHead == NULL)
    {
        cout << "\nDS don vong rong!\n";
        return;
    }
    p = l.pHead;
    do
```

```

    {
        cout << p->info << '\t';
        p = p->pNext;
    } while (p != l.pHead); //Chua giap vong
}
//Tap tin -> LLIST
int File_LLIST(char *f, LLIST &l)
{
    ifstream in(f);    //Mo de doc
    if (!in)
        return 0;

    CreatLLIST(l);
    data x;
    in >> x;
    InsertTail(l, x);
    while (!in.eof())
    {
        in >> x;
        InsertTail(l, x);
    }
    in.close();
    return 1;
}

- Tập tin thuvien.h :
//so ngau nhien n
int TaoSoNgauNhien()
{
    int n;
    srand(time(NULL));
    n = rand() % MAX + 1;
    return n;
}
//ham tra ve con tro tro den nut thu n-1
NODE *Node_q(LLIST l, int n)
{
    NODE *q;
    int dem = 1; //dem nut da qua
    q = l.pHead;
    if (q != NULL)
    {
        while (dem < n-1)
        {
            q = q->pNext;
            dem++;
        }
    }
    return q;
}
//Huy nut sau q : huy xong, chuyen nut ke tiep la nut dau, q la cuoi
void Remove_Node_After_q(LLIST &l, NODE *q)
{
    if (l.pHead == NULL)
    {
        cout << "\nDanh sach rong!";
        _getch();
        return;
    }

```



```

NODE *p = q->pNext;
q->pNext = p->pNext;
delete p;

l.pHead = q->pNext;
if (l.pHead == NULL) //con lai rong
    l.pTail = NULL;
else
if (l.pHead != l.pTail) //khong phai con lai 1 nut
    l.pTail = q; //cap nhat tail
}
//dem nut
int SoNut(LLIST l)
{
    int dem;
    NODE *p = l.pHead;
    if (p == NULL)
        dem = 0;
    else
    {
        dem = 1;
        while (p != l.pTail)
        {
            dem++;
            p = p->pNext;
        }
    }
    return dem;
}
//Xuat cac binh si ra khoi vong, binh si chon di cau cuu
void CryForHelp(LLIST l)
{
    int n = TaoSoNgauNhien();
    int sn = SoNut(l);
    NODE *q;
    int lan = 0;
    cout << "\nSo ngau nhien: n = " << n;
    cout << "\n          KET QUA TUYEN CHON:\n";
    while (sn > 1)
    {
        q = Node_q(l, n); //tro den nut thu n-1
        lan++;
        cout << setiosflags(ios::left);
        cout << setw(21) << "\nRa khoi vong lan thu " << setw(4) << lan << setw(13)
            << " la binh si : " << q->pNext->info;
        Remove_Node_After_q(l, q);
        sn--;
    }
    cout << "\n\n          Binh si duoc cu di cau cuu la : " << l.pHead->info;
}

```

LAB 08. Cây nhị phân - cây nhị phân tìm kiếm

A. Mục tiêu

- Tìm hiểu tổ chức cây nhị phân (Binary Tree : BT)
- Tìm hiểu tổ chức cây nhị phân tìm kiếm (Binary Search Tree : BST)
- Thực hiện các thao tác cơ bản trên cây nhị phân tìm kiếm
- Tìm hiểu các ứng dụng liên quan

B. Yêu cầu

- Phần ôn tập (C) : sinh viên tự ôn.

- **Buổi thực hành 9** (3 t)

- Sinh viên thực tập trong phòng lab :
 - Khối lượng và nội dung : **Các bài trong phần D (luyện tập)**
 - Yêu cầu lưu trữ
 - ✓ Tạo thư mục **MaSV_HoVaTen_Lab08_BST** chứa trong ổ đĩa qui định.
 - ✓ Mỗi bài, tạo project lưu trữ trong thư mục trên.
 - ✓ Xóa thư mục debug trong mỗi project.
 - ✓ Nén thư mục
 - Hình thức thu bài
 - ✓ Hệ thống thu bài trong phòng lab.

C. Ôn tập

I. Tổ chức kiểu dữ liệu BST:

Cây nhị phân tìm kiếm (BST) là cây nhị phân mà khoá tại mỗi nút cây lớn hơn khoá của tất cả các nút thuộc cây con bên trái và nhỏ hơn khoá của tất cả các nút thuộc cây con bên phải.

Ta dùng thuật ngữ khó hàm ý dữ liệu của các nút là có thứ tự (so sánh được với nhau). Nếu dữ liệu của nút là một cấu trúc có nhiều thành phần, ta quan tâm tới thành phần có thứ tự, là trường khóa của cấu trúc.

II. Cài đặt kiểu dữ liệu cây nhị phân tìm kiếm :

- Kiểu của thành phần dữ liệu trong các nút của cây (giả sử là int):
typedef int KeyType;
- Kiểu các nút trong cây :

```
struct BSNode
{
    KeyType key;
    BSNode *left; //Chưa địa chỉ cây con trái
    BSNode *right; //Chưa địa chỉ cây con phải
};
```

- Kiểu cây nhị phân tìm kiếm
(là kiểu con trỏ trỏ đến các phần tử kiểu BSNode)

```
typedef BSNode *BSTree;
```

III. Các thao tác thường dùng trên cây nhị phân tìm kiếm

(Giả sử ta đã cài đặt kiểu BST như trên)

1. Tạo nút mới: CreateNode (x) :

- Chức năng: Tạo ra một phần tử (nút) của BST với thành phần dữ liệu là x
(Các thành phần liên kết của nút mới là con trỏ có giá trị NULL)
- Input : x
- Output : trả về con trỏ - lưu trữ địa chỉ của pt vừa tạo (nếu tạo thành công) - có giá trị NULL (nếu ngược lại).

- Nguyên mẫu của hàm: `BSNode* GetNode(KeyType x);`
2. Khởi tạo cây BST rỗng: **CreateBST (l)**
- Chức năng: Tạo ra một BST rỗng.
 - Input :
 - Output : root rỗng
 - Nguyên mẫu của hàm: `void CreateBST(BSTree &root);`
3. Duyệt cây:
- Chức năng: Duyệt qua các nút của cây, mỗi nút 1 lần
 - Có 3 phương thức duyệt : tiền tự, trung tự, hậu tự . Tại mỗi nút, xử lý dữ liệu như thế nào tùy theo yêu cầu bài toán. Nguyên mẫu của hàm cho các phương thức :
 - Tiền tự (N-L-R)
`void PreOrder(BSTree root);`
 - Trung tự (L-N-R)
`void InOrder(BSTree root);`
 - Hậu tự (L-R-N)
`void PosOrder(BSTree root);`
4. Tìm nút có key là x:
- Chức năng: Tìm trong BST có nút chứa key là x?
 - Input : root, x
 - Output : p (con trỏ p chứa nút key là x); nếu có NULL; ngược lại
 - Nguyên mẫu của hàm: `BSTree Search(KeyType x, BSTree root);`
5. Thêm x vào BST:
- Chức năng: Thêm một key vào BST
 - Input :root, x
 - Output : -1; không thành công - không đủ bộ nhớ
0; không thành công - x đã có sẵn trong cây
1; Thành công; cây BST mới có thêm nút chứa x
 - Nguyên mẫu của hàm: `int insertNode(BSTree &root, KeyType x);`
6. Hủy nút có khóa nhỏ nhất của cây con phải của nút
- Chức năng: Hủy nút cực trái của cây con phải của nút (sẽ hủy)
 - Input : root
 - Output : root (đã xóa được nút có key nhỏ nhất của cây con phải) root x; khóa của nút bị xóa
 - Nguyên mẫu của hàm: `KeyType DeleteMin(BSTree &root);`
7. Hủy nút có key là x cho trước :
- Chức năng: Hủy nút có key x.
 - Input root, x
 - Output : 1, root; cây BST kết quả, đã xóa được nút có x; nếu thành công
0; không thành công
 - Nguyên mẫu của hàm: `int DeleteNode(KeyType x, BSTree &root);`
8. Hủy cây BST :
- Chức năng: Hủy tất cả các nút của cây BST
 - Input : root
 - Output : root rỗng
 - Nguyên mẫu của hàm: `void RemoveTree(BSTree &root);`
9. Tạo cây BST từ tập tin
- Chức năng: Chuyển dữ liệu tập tin filename vào BST.
 - Input : filename
 - Output : 1; chuyển dữ liệu vào BST root thành công
0; không thành công

- Nguyên mẫu của hàm : `int File_BST(BSTree &root, char *filename);`

10. Xuất BST ra màn hình theo thứ tự trước, giữa, cuối.

D. LUYỆN TẬP

Bài 1:

Viết chương trình tùy chọn thực hiện các thao tác cơ bản trên cây nhị phân tìm kiếm, khóa các nút trong là số nguyên:.

1. Tạo cây nhị phân tìm kiếm
2. Xem cây nhị phân tìm kiếm theo thứ tự trước, giữa, sau
3. Tính số nút của cây
4. Đếm số nút có khóa < x cho trước và xuất các nút ra màn hình.
5. Tìm nút có khóa x cho trước
6. Kiểm tra nút có khóa cho trước có phải là nút lá
7. Đếm số nút lá của cây
8. Xác định chiều cao của cây
9. Xác định mức của nút có khóa x cho trước
10. So sánh mức của 2 nút
11. Thêm một khóa x vào cây nhị phân tìm kiếm
12. Hủy nút có khóa x cho trước.

Dữ liệu được cho trong các tập tin :

- "Test1.txt" :

25 37 10 18 29 50 3 1 6 5 12 20 35 13 32 41

- "Test2.txt"

44 18 13 15 37 23 88 59 55 71 108

Thực hiện :

Tạo thư mục **MaSV_HoVaTen_Lab08** để chứa các project

- Bước 1. Tạo project win32 console application : Bai1_D, với các tập tin :

- Header.h : Chứa định nghĩa dữ liệu, các khai báo nguyên mẫu các hàm hệ thống, chức năng, định nghĩa các hàm hệ thống, nhập xuất.
- ThuVien.h : Chứa các định nghĩa các hàm chức năng
- Menu.h : Chứa các hàm tổ chức menu
- Program.cpp : Chứa main() thực hiện chương trình

- Bước 2 : Soạn thảo phần code tối thiểu trong Program.cpp chạy được chương trình :

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <conio.h>
```

```
using namespace std;
```

```
#include "Header.h"
```

```
#include "Thuvien.h"
```

```
#include "Menu.h"
```

```
void ChayChuongTrinh();
```

```
int main()
```

```
{
```

```
    ChayChuongTrinh();
```

```
    return 1;
```

```
}
```

```
void ChayChuongTrinh()
```

```
{
```

```
    System("PAUSE");
```

```
}
```

- Bước 3 :

Trong Header.h, định nghĩa cấu trúc dữ liệu BST, các hàm hệ thống và nhập, xuất (tạo nút, tạo BST rỗng, thêm khóa vào BST, Nhập cây, Xuất cây theo thứ tự)

- Nội dung tập tin Header.h :

```
//Định nghĩa Cấu trúc dữ liệu : BST
typedef int KeyType; //Kiểu dữ liệu của thành phần dữ liệu của các nút trong cây là int

//Kiểu các nút của cây BST: BSNODE
struct BSNODE
{
    KeyType key;
    BSNODE *left; //Chứa địa chỉ cây con trái
    BSNODE *right; //Chứa địa chỉ cây con phải
};

//Kiểu CTDL Cây nhị phân tìm kiếm : kiểu con trỏ trỏ đến các nút kiểu BSNODE
typedef BSNODE *BSTree;

//=====
//Khai báo nguyên mẫu các hàm
//Các hàm hệ thống

void CreateBST(BSTree &root); //Tạo cây BST rỗng
BSNODE *CreateNode(KeyType x);

//Chèn
int InsertNode(BSTree &root, KeyType x); //Chèn x vào cây BST

//Nhập, xuất (duyet va in)
int File_BST(BSTree &root, char *filename); //Tạo cây BST từ file
void PreOrder(BSTree root); //NLR
void InOrder(BSTree root); //LNR
void PosOrder(BSTree root); //LRN

//=====
//Định nghĩa hàm

//Tạo nút với x cho trước
BSNODE *CreateNode(KeyType x)
{
    BSNODE *p = new BSNODE;
    if (p != NULL)
    {
        p->key = x;
        p->left = NULL;
        p->right = NULL;
    }
    return p;
}

//-----
//Khởi tạo cây BST rỗng
void CreateBST(BSTree &root)
{
    root = NULL;
}
```

```
//-----  
//Them x vào cây BST  
int InsertNode(BSTree &root, KeyType x)  
{  
    //Cây khác rỗng  
    if (root != NULL)  
    {  
        if (root->key == x)  
            return 0; // x đã có sẵn  
        if (root->key > x)  
            return InsertNode(root->left, x);  
        else  
            return InsertNode(root->right, x);  
    } //root == NULL  
  
    root = CreateNode(x);  
    if (root == NULL)  
        return -1;  
    return 1;  
}  
//-----  
//Tạo cây BST từ file  
int File_BST(BSTree &root, char *filename)  
{  
    ifstream in(filename);  
    if (!in)  
        return 0;  
    KeyType x;  
    int kq;  
    CreateBST(root);  
    in >> x;  
    kq = InsertNode(root, x);  
    if (kq == 0 || kq == -1)  
        return 0;  
    while (!in.eof())  
    {  
        in >> x;  
        kq = InsertNode(root, x);  
        if (kq == 0 || kq == -1)  
            return 0;  
    }  
    in.close();  
    return 1;  
}  
//-----  
//Xuất cây theo thứ tự trước : NLR  
void PreOrder(BSTree root)  
{  
    if (root != NULL)  
    {  
        cout << root->key << "t";  
        PreOrder(root->left);  
        PreOrder(root->right);  
    }  
}  
  
//Xuất cây theo thứ tự giữa : LNR
```

```
void InOrder(BSTree root)
{
    if (root != NULL)
    {
        InOrder(root->left);
        cout << root->key << '\t';
        InOrder(root->right);
    }
}
```

//Xuất cây theo thu tự sau : LRN

```
void PosOrder(BSTree root)
{
    if (root != NULL)
    {
        PosOrder(root->left);
        PosOrder(root->right);
        cout << root->key << '\t';
    }
}
//-----
```

- Bước 4 :

Trong menu.h, soạn thảo các hàm tổ chức menu :

```
void XuatMenu()
{
    cout << "\n=====He thong menu=====";
    cout << "\n0. Thoat khoi chuongbg trinh";
    cout << "\n1. Tao cay BST";
    cout << "\n2. Xuat cay BST theo thu tu truoc, giua, cuoi";
    cout << "\n3. So nut cua cay";
    cout << "\n4. Dem so nut co key <x";
    cout << "\n5. Tim nut co khoa x";
    cout << "\n6. Kiem tra nut co key cho truoc co phai la nut la";
    cout << "\n7. Dem so nut la va xuat cac nut la";
    cout << "\n8. Chieu cao cua cay";
    cout << "\n9.Muc cua nut co khoa x";
    cout << "\n10.So sanh muc 2 nut";
    cout << "\n11. Them khoa x vao cay";
    cout << "\n12. Xoa nut co khoa x";
}
```

```
int ChonMenu(int soMenu)
{
    //...
}
```

```
void XuLyMenu(int menu, BSTree &root)
{
    switch (menu)
    {
        case 0:
            system("CLS");
            cout << "\n0. Thoat khoi chuong trinh\n";
            break;
        case 1:
            system("CLS");
            cout << "\n1. Tao cay BST";
```

```
        break;
    case 2:
        system("CLS");
        cout << "\n2. Xuất cây BST theo thu tu";
        break;
    //...
}
}
```

- Bước 5 : Vận hành hệ thống menu

Trong Program.cpp, cập nhật lại hàm ChayChuongTrinh() như sau :

```
void ChayChuongTrinh()
{
    int menu, soMenu = 13;
    BSTree root = NULL;
    do
    {
        menu = ChonMenu(soMenu);
        XuLyMenu(menu, root);
        system("PAUSE");
    } while (menu > 0);
}
```

- Bước 6 :

- Tạo các tập tin dữ liệu :test1.txt, test2.txt theo yêu cầu bài toán.
- Cập nhật lại hàm XuLyMenu :
 - sử dụng hàm tạo cây từ tập tin trong case 1
 - Sử dụng các hàm xuất cây theo thứ tự trong case2
- **Nội dung hàm XuLyMenu :**

```
void XuLyMenu(int menu, BSTree &root)
{
    char *filename;
    int kq ;

    //=====
    switch (menu)
    {
        case 0:
            system("CLS");
            cout << "\0. Thoat khỏi chuong trình\n";
            break;

        case 1:
            system("CLS");
            cout << "\n1. Tao cay BST";
            filename = new char[50];
            cout << "\nTen tap tin : filename = ";
            _flushall();
            cin >> filename;
            kq = File_BST(root, filename);
            if (kq)
                cout << "\nDa chuyen du lieu file " << filename << " vao cay BST";
            else
                cout << "\nKhong thanh cong!!!";
            break;

        case 2:
```



```
system("CLS");
cout << "\n2. Xuất cây BST theo thu tu giữa";
cout << "\n\nCây BST hiện hành, xuất theo thu tu trước (NLR) :\n";
PreOrder(root);
cout << "\n\nCây BST hiện hành, xuất theo thu tu giữa (LNR) :\n";
InOrder(root);
cout << "\n\nCây BST hiện hành, xuất theo thu tu cuối (LRN) :\n";
PosOrder(root);
break;
//. . .
}
}
```

- Từ bước 7 trở về sau cho đến xét hết các chức năng chương trình :

- Trong tập tin THuVien.h , soạn các hàm chức năng của chương trình
- Trong hàm XuLyMenu của menu.h, bổ sung sử dụng các hàm chức năng tương ứng vào các case trong câu lệnh lựa chọn switch.

- Tham khảo nội dung tập tin **ThuVien.h** :

//Khai báo nguyên mẫu

```
int DemSoNut(BSTree root);
int DemNutNhoHon(BSTree root, KeyType x);
BSTree Search(KeyType x, BSTree root);
int IsLeaf_x(BSTree root, KeyType x);
int DemNutLa(BSTree root);
int TinhMax(int a, int b);
int TinhChieuCao(BSTree root);
int TimMuc_x(BSTree root, KeyType x);
KeyType DeleteMin(BSTree &root);
int DeleteNode(KeyType x, BSTree &root);
void RemoveTree(BSTree &root);
```

//Định nghĩa hàm

//Dem so nut

```
int DemSoNut(BSTree root)
{
    if (root == NULL)
        return 0;
    return 1 + DemSoNut(root->left) + DemSoNut(root->right);
}
```

//Tinh so nut < x va xuất các nut ra màn hình

```
int DemNutNhoHon(BSTree root, KeyType x)
{
    int soNut;
    if (root == NULL)
        soNut = 0;
    else
        if (root->key < x)
        {
            cout << root->key << "\t";
            soNut = 1 + DemNutNhoHon(root->left, x) + DemNutNhoHon(root->right, x);
        }
    else
        soNut = DemNutNhoHon(root->left, x) + DemNutNhoHon(root->right, x);
}
```

```

        return soNut;
    }

//Tim kien nut co khoa x cho truoac
BSTree Search(KeyType x, BSTree root)
{
    if (root != NULL)
    {
        if (root->key == x) //Tim thay x
            return root;
        else
            if (root->key < x)
                return Search(x, root->right); //tim x trong cay con phai
            else
                return Search(x, root->left); //tim x trong cay con trai
    }
    return NULL; //khong co
}

//Kiem tra nut co khoa cho truoac co phai la nut la
int IsLeaf_x(BSTree root, KeyType x)
{
    int kq = 0; //khong la nut la
    BSTree T = Search(x, root);
    if (T == NULL)
        kq = -1; //x khong co trong cay
    else
        kq = (T->left == NULL) && (T->right == NULL);
    return kq;
}

//Dem so nut la
int DemNutLa(BSTree root)
{
    int soNutLa;
    if (root == NULL)
        soNutLa = 0;
    else
        if (root->left == NULL && root->right == NULL)
        {
            cout << root->key << "t";
            soNutLa = 1;
        }
        else
            soNutLa = DemNutLa(root->left) + DemNutLa(root->right);

    return soNutLa;
}

int TinhMax(int a, int b)
{
    if (a >= b)
        return a;
    return b;
}

//Chieu cao cua cay
int TinhChieuCao(BSTree root)

```

```
{
    int h;
    if (root == NULL)
        h = -1;
    else
        if (root->left == NULL && root->right == NULL)
            h = 0;
        else
            h = 1 + TinhMax(TinhChieuCao(root->left), TinhChieuCao(root->right));
    return h;
}
```

//Tim Muc cua nut co khoa x

//Muc goc = 0

int TimMuc_x(BSTree root, KeyType x)

```
{
    int muc;
    muc = 0;
    BSTree T = root;
    while (T != NULL)
    {
        if (T->key == x)
            break;
        else
        {
            muc++;
            if (T->key > x)
                T = T->left;
            else
                T = T->right;
        }
    }
    return muc;
}
```

//So sanh muc 2 nut : cung muc hay kho khac

int SoSanhMuc(BSTree root, KeyType x, KeyType y)

```
{
    int kq1, kq2;
    kq1 = TimMuc_x(root, x);
    kq2 = TimMuc_x(root, y);
    if (kq1 > kq2)
        return 1;
    else
        if (kq1 < kq2)
            return -1;
        else
            return 0;
}
```

//Huy nut co gia tri nho nhat cua cay con phai cua root : nut cuc trai

//Input root

//Output : root (da xoa duoc nut co gia tri nho nhat cua cay con trai root

// x; khoa cua nut bi xoa

KeyType DeleteMin(BSTree &root)

```
{
    KeyType k;
    if (root->left == NULL)
    {
        k = root->key;
        root = root->right;
    }
}
```

```

        return k;
    }
    else
        return DeleteMin(root->left);
}
//Hủy một nút có khóa cho trước ra khỏi cây
//Input : x, root
//Output : 1, root (cây BST kết quả qua root) nếu thành công
//        0; không thành công

int DeleteNode(KeyType x, BSTree &root)
{
    if (root != NULL)
    {
        if (x < root->key)
            DeleteNode(x, root->left);
        else
            if (x > root->key)
                DeleteNode(x, root->right);
            else //x == root->key
                if ((root->left == NULL) && (root->right == NULL))
                    //không có con trái, không có con phải
                    root = NULL;
                else
                    if (root->left == NULL)
                        //có 1 con : con phải, không có con trái
                        root = root->right;
                    else
                        if (root->right == NULL)
                            //có 1 con : con trái, không có con phải
                            root = root->left;
                        else //có cả 2 con trái, phải
                            root->key = DeleteMin(root->right);

        return 1; //Thành công
    }
    return 0; //không thành công
}

```

- Tham khảo hàm **XuLyMenu** :

```

void XuLyMenu(int menu, BSTree &root)
{
    char *filename;
    int kq;
    KeyType x, y;

    //=====
    switch (menu)
    {
        case 0:
            system("CLS");
            cout << "\0. Thoát khỏi chương trình\n";
            break;
        case 1:
            system("CLS");
            cout << "\n1. Tạo cây BST";
            filename = new char[50];
            cout << "\nTen tap tin : filename = ";

```

```
_flushall();
cin >> filename;
kq = File_BST(root, filename);
if (kq)
    cout << "\nĐã chuyển dữ liệu file " << filename << " vào cây BST";
else
    cout << "\nKhông thành công!!!";

delete[]filename;
break;
case 2:
    system("CLS");
    cout << "\n2. Xuất cây BST theo thứ tự giữa";
    cout << "\nCây BST hiện hành, xuất theo thứ tự trước (NLR) :\n";
    PreOrder(root);
    System("PAUSE");

    cout << "\nCây BST hiện hành, xuất theo thứ tự giữa (LNR) :\n";
    InOrder(root);
    System("PAUSE");

    cout << "\nCây BST hiện hành, xuất theo thứ tự cuối (LRN) :\n";
    PosOrder(root);
    System("PAUSE");

    break;
case 3:
    system("CLS");
    cout << "\n3. Số nút của cây";
    cout << "\nCây BST hiện hành, xuất theo thứ tự giữa :\n";
    InOrder(root);
    cout << "\nSố nút trong cây : Số nút = " << DemSoNut(root);
    cout << endl;
    break;
case 4:
    system("CLS");
    cout << "\n4. Đếm số nút có key < x và xuất các nút ra màn hình";
    cout << "\nCây BST hiện hành, xuất theo thứ tự giữa :\n";
    InOrder(root);
    cout << "\nx = ";
    cin >> x;
    cout << "\nDanh sách các nút của BST có key < " << x << " :\n";
    kq = DemNutNhoHon(root, x);
    cout << "\nSố nút của cây BST có key < " << x << " : " << kq;
    cout << endl;
    break;
case 5:
    system("CLS");
    cout << "\n5. Tìm kiếm khóa x";
    cout << "\nCây BST hiện hành, xuất theo thứ tự giữa :\n";
    InOrder(root);
    cout << endl;

    cout << "\nTìm khóa x = ";
    cin >> x;
    if (Search(x, root))
        cout << "\nkhóa x = " << x << " có trong cây.\n";
```

```
else
    cout << "\nkhoa x = " << x << " không có trong cây!\n";
break;
```

case 6:

```
system("CLS");
cout << "\n6. Kiểm tra nút cho trước có phải là nút lá";
cout << "\nCây BST hiện hành, xuất theo thứ tự giữa :\n";
InOrder(root);
cout << "\nx = ";
cin >> x;
system("CLS");
kq = IsLeaf_x(root, x);
if (kq == -1)
    cout << "\n\nkey " << x << " không có trong cây ";
else
    if (kq == 0)
        cout << "\n\nNut có key " << x << " không là nút lá ";
    else
        cout << "\n\nNut có key " << x << " là nút lá ";
cout << endl;
break;
```

case 7:

```
system("CLS");
cout << "\n7. Đếm số nút lá và xuất các nút lá";
cout << "\nCây BST hiện hành, xuất theo thứ tự giữa :\n";
InOrder(root);
cout << "\n\nDanh sách các nút lá của BST:\n";
kq = DemNutLa(root);

cout << "\n\nSố nút lá của cây BST : soNutLa = " << kq;
cout << endl;
break;
```

case 8:

```
system("CLS");
cout << "\n8. Chiều cao của cây";
cout << "\nCây BST hiện hành, xuất theo thứ tự giữa :\n";
InOrder(root);
cout << "\nChiều cao của cây BST : h = " << TinhChieuCao(root);
cout << endl;
break;
```

case 9:

```
system("CLS");
cout << "\n9. Mức của nút có khóa x";
cout << "\nCây BST hiện hành, xuất theo thứ tự giữa :\n";
InOrder(root);
cout << "\nx = ";
cin >> x;
if (!Search(x, root))
    cout << "\nCây rỗng hoặc Không có nút nào có khóa " << x;
else
    cout << "\n\nMức của nút có khóa " << x << " : " << TimMuc_x(root, x);
cout << endl;
break;
```

case 10:

```
system("CLS");
cout << "\n10. So sánh mức 2 nút";
cout << "\nCây BST hiện hành, xuất theo thứ tự giữa :\n";
```

```
InOrder(root);
cout << "\nNhap khoa x : ";
cin >> x;
cout << "\nNhap khoa y : ";
cin >> y;
if (SoSanhMuc(root, x, y) == 1)
    cout << "\n\nMuc của nút có khoa " << x << " sau hơn mức của nút có khoa " << y;
else
if (SoSanhMuc(root, x, y) == -1)
    cout << "\n\nMuc của nút có khoa " << y << " sau hơn mức của nút có khoa " << x;
else
    cout << "\n\nNut có khoa " << x << " và nút có khoa " << y << " cùng mức";
_getch();
cout << "\n===== ";
cout << "\n\nMuc của nút có khoa " << x << " : " << TimMuc_x(root, x);
cout << "\n\nMuc của nút có khoa " << y << " : " << TimMuc_x(root, y);
break;
case 11:
system("CLS");
cout << "\n11. Thêm khoa x vào cây";
cout << "\nCây BST hiện hành, xuất theo thứ tự giữa :\n";
InOrder(root);
cout << "\nKhoa x = ";
cin >> x;
if (InsertNode(root, x) != 1)
    cout << "\nThêm khoa " << x << " không thành công\n";
else
{
    cout << "\nCây BST sau khi thêm khoa " << x << " :\n";
    InOrder(root);
}
cout << endl;
break;
case 12:
system("CLS");
cout << "\n12. Xóa nút có khoa x";
cout << "\nCây BST hiện hành, xuất theo thứ tự giữa :\n";
InOrder(root);
cout << "\nKhoa x = ";
cin >> x;

if (!Search(x, root))
{
    cout << "\nkhoa x = " << x << " không có trong cây.\n";
    _getch();
    break;
}
else
if (DeleteNode(x, root) != 0)
{
    cout << "\nCây BST sau khi xóa khoa " << x << " :\n";
    InOrder(root);
}
cout << endl;
break;
}
```

Bài 2:

Viết chương trình tùy chọn thực hiện các thao tác cơ bản trên cây nhị phân tìm kiếm, khóa các nút trong là số nguyên..

1. Tạo cây tìm kiếm nhị phân
2. Xuất cây tìm kiếm nhị phân theo thứ tự trước, giữa, sau
3. Xuất cây theo chiều sâu
4. Xuất cây theo chiều rộng
5. Xuất cây theo từng mức

Dữ liệu cho như bài 1.

Thực hiện :

Tạo project win32 console application : Bai2_D, với các tập tin :

- Bstree.h : Chứa định nghĩa dữ liệu, các hàm liên quan đến BST
- List.h : Chứa định nghĩa dữ liệu, các hàm liên qua đến List (Stack, Queue) và các chức năng bài toán : xuất theo chiều sâu, chiều rộng
- Program.cpp : Chứa hàm main() thực hiện chương trình.

- Tham khảo tập tin "BSTree.h" :

```
//Định nghĩa Cấu trúc dữ liệu :BST
typedef int KeyType; //Kiểu dữ liệu của thành phần dữ liệu của các nút trong cây là int

//Kiểu các nút của cây : BSNode
struct BSNode
{
    KeyType key;
    BSNode *left; //Chứa địa chỉ cây con trái
    BSNode *right; //Chứa địa chỉ cây con phải
};

//Kiểu CSDL Cây nhị phân tìm kiếm : kiểu con trỏ trỏ đến các nút kiểu BSNode
typedef BSNode *BSTree;

//=====
//Khai báo nguyên mẫu các hàm
//Các hàm hệ thống

void CreateBST(BSTree &root); //Tạo cây BST rỗng
BSNode *CreateNode(KeyType x);

//Chen
int InsertNode(BSTree &root, KeyType x); //Chen x vào cây BST

//Nhập, xuất (duyet va in)
int File_BST(BSTree &root, char *filename); //Tạo cây BST từ file
void PreOrder(BSTree root); //NLR
void InOrder(BSTree root); //LNR
void PosOrder(BSTree root); //LRN
KeyType TinhMax(KeyType a, KeyType b);
int TinhChieuCao(BSTree root);
int TimMuc_x(BSTree root, KeyType x);
```

(Xem bài 1)

- Tham khảo tập tin "List.h" :

//List <-> Queue + Stack

//Kiểu thành phần dữ liệu của các nút trong Queue, stack là BSNode* <-> BSTree


```
//Kieu cac nut trong Queue, Stack
struct NODE
{
    BSTree info;
    NODE* pNext;
};

struct List //Queue, Stack
{
    NODE *pHead;
    NODE *pTail;
};

//Tao nut rong
NODE *GetNode(BSTree info)
{
    NODE *p = new NODE;
    if (p != NULL)
    {
        p->info = info;
        p->pNext = NULL;
    }
    return p;
}

//Tao List rong (Queue, Stack)
void CreateList(List &l)
{
    l.pHead = l.pTail = NULL;
}

int IsEmpty(List l)
{
    if (l.pHead == NULL) return 1;
    return 0;
}

//Them cuoi : EnQueue -> phục vụ cho Queue
void Inserttail(List &l, BSTree info)
{
    NODE *x = GetNode(info);
    if (x == NULL)
        return;
    if (l.pHead == NULL)
    {
        l.pHead = x;
        l.pTail = l.pHead;
    }
    else
    {
        l.pTail->pNext = x;
        l.pTail = x;
        l.pTail->pNext = NULL;
    }
}
```

//Xoa dau, tra ve du lieu nut bi xoa : ->Phuc vu cho Queue, Stack : DeQueue , Pop
BSTree RemoHead(List &l)//lay ra va huy ptu dau list.

```
{  
    NODE *p;  
    BSTree x;  
    if (l.pHead == NULL)  
        return NULL;  
    else  
    {  
        p = l.pHead;  
        x = p->info;  
        l.pHead = p->pNext;  
        delete p;  
        return x;  
    }  
}
```

//Them dau, phuc vu cho Stack : Push
void InsertHead(List &s, BSTree info)

```
{  
    NODE *x = GetNode(info);  
    if (x == NULL)  
        return;  
    if (s.pHead == NULL)  
    {  
        s.pHead = x;  
        s.pTail = s.pHead;  
    }  
    else  
    {  
        x->pNext = s.pHead->pNext;  
        s.pHead->pNext = x;  
        s.pHead = x;  
    }  
}
```

//-----
// Queue
//-----

//Xuat theo chieu rong (tung muc) : Dung Queue

```
void BFS(BSTree root)  
{  
    List q;  
    NODE *t;  
    BSTree p;  
    CreateList(q);  
    if (root == NULL)  
        cout << "\n Cay rong.";  
    else  
    {  
        t = GetNode(root);  
        Inserttail(q, t->info);  
        do  
        {  
            p = RemoHead(q);  
            cout << p->key << '\t';  
            if (p->left != NULL)  
            {  
                t = GetNode(p->left);
```

```

        Inserttail(q, t->info);
    }
    if (p->right != NULL)
    {
        t = GetNode(p->right);
        Inserttail(q, t->info);
    }

    } while (IsEmpty(q) == 0);
    cout << endl;
}

//Xuat nut muc k
void BFS_Muc_k(BSTree root, int k)
{
    List q;
    NODE *t;
    BSTree p;
    CreateList(q);
    if (root == NULL)
        cout << "\n Cay rong.";
    else
    {
        t = GetNode(root);
        Inserttail(q, t->info);
        do
        {
            p = RemoHead(q);
            if (TimMuc_x(root, p->key) == k)
                cout << p->key << "t";
            if (p->left != NULL)
            {
                t = GetNode(p->left);
                Inserttail(q, t->info);
            }
            if (p->right != NULL)
            {
                t = GetNode(p->right);
                Inserttail(q, t->info);
            }
        } while (IsEmpty(q) == 0);
        cout << endl;
    }
}

//Xuat theo muc
void BFS_TheoMuc(BSTree root)
{
    int i, h;
    h = TinhChieuCao(root);
    cout << "\n\nXuat BST theo chieu rong (tung muc):\n";
    for (i = 0; i <= h; i++)
    {
        cout << endl << "Muc " << i << " : ";
        BFS_TheoMuc_k(root, i);
    }
}

//-----

```

```
//      Stack
//-----
//Xuat theo chieu sau : dung Stack
void DFS(BSTree root)
{
    List s;
    NODE *t;
    BSTree p;
    CreateList(s);
    if (root == NULL)
        cout << "\n Cay rong.";
    else
    {
        t = GetNode(root);
        InsertHead(s, t->info);
        do
        {
            p = RemoHead(s);
            cout << p->key << '\t';

            if (p->left != NULL)
                DFS(p->left);
            if (p->right != NULL)
            {
                t = GetNode(p->right);
                InsertHead(s, t->info);
            }
        } while (IsEmpty(s) == 0);
        cout << endl;
    }
}
- Tham khảo hàm ChayChuongTrinh() trong "Program.cpp":
void ChayChuongTrinh()
{
    char *filename;
    filename = new char[50];
    BSTree root;
    int kq;
    do
    {
        cout << "\nFilename : ";
        cin >> filename;
        kq = File_BST(root, filename);
    } while (!kq);
    cout << "\n\nXuat BST theo thu tu truoc:\n";
    PreOrder(root);
    cout << "\n\nXuat BST theo thu tu giua:\n";
    InOrder(root);
    cout << "\n\nXuat BST theo thu tu sau:\n";
    PosOrder(root);
    cout << "\n\nXuat BST theo chieu sau :\n";
    DFS(root);
    system("PAUSE");
    cout << "\n\nXuat BST theo chieu rong :\n";
    BFS(root);
    system("PAUSE");
    BFS_TheoMuc(root);
    system("PAUSE");
}
```

Bài 3.

Bảng lương nhân viên của một công ty chứa các thông tin của nhân viên như sau :

- Mã nhân viên, //chuỗi có đúng 7 ký tự, không có ký tự trắng
- Họ và Tên của nhân viên, // chuỗi có không quá 25 ký tự
- Năm sinh, số nguyên dương có 4 ký số
- Địa chỉ, // chuỗi có không quá 15 ký tự
- Lương //số thực dương

Viết chương trình tùy chọn thực hiện trên bảng lương nhân viên, với bảng lương được cài đặt bằng cây nhị phân tìm kiếm với khóa là mã nhân viên :

8. Thoát khỏi chương trình
9. Tạo bảng lương nhân viên (chuyển dữ liệu từ tập tin cho trước vào cây nhị phân tìm kiếm dựa vào khóa mã nhân viên.
10. Xem bảng lương nhân viên theo thứ tự trước (NLR), thứ tự giữa (LNR), thứ tự cuối (LRN)
11. Tính số nút của cây.
12. Thay đổi năm sinh của nhân viên khi biết mã nhân viên
13. Hủy nhân viên có mã nhân viên cho trước ra khỏi bảng lương.

- Tập tin dữ liệu **“bl.txt”** lưu trữ thông tin các nhân viên trong bảng lương với định dạng như sau :
 - Tập tin có 12 hàng, mỗi hàng gồm 5 cột (các cột tách biệt bằng ký tự tách như ký tự trắng, tab,...) chứa các thông tin một nhân viên : Cột 1 chứa mã nhân viên, cột 2 chứa Họ và Tên nhân viên, cột 3 chứa năm sinh (NS) nhân viên, cột 4 chứa địa chỉ nhân viên, cột cuối cùng chứa lương nhân viên.
 - Các mã nhân viên đôi một khác nhau. Mã nhân viên xác định duy nhất 1 nhân viên.
 - Các chuỗi lưu trữ thông tin về họ và tên nhân viên, địa chỉ của nhân viên có thể gồm nhiều từ, các từ được nối với nhau bởi dấu gạch dưới.

Mã NV	Họ và Tên nhân viên	NS	Địa chỉ	Lương
LD22145	Le_Minh	1986	Khanh_Hoa	12000000
LD12045	Nguyen_Tuan_Vo	1980	Lam_Dong	30000000
LD15332	Le_Thi_Hoa	1974	Binh_Dinh	10000000
LD23045	Tran_Le_Trong	1991	Ha_Noi	25000000
LD13452	Tran_Van_Ngoc	1974	Khanh_Hoa	30000000
LD13210	Ly_Van_Hoa	1985	Ninh_Thuan	30000000
LD24042	Vo_Ngoc_Hoa	1983	Ha_Noi	30000000
LD14432	Nguyen_Vo	1985	Phu_Yen	12000000
LD22032	Van_Thi_Hanh	1984	Lam_Dong	10000000
LD22140	Tran_Minh_Vuong	1990	Binh_Dinh	12000000
LD30432	Nguyen_Vo	1975	Lam_Dong	10000000
LD22052	Vo_Ngoc_Thai	1985	Lam_Dong	10000000

Thực hiện :

- Bước 1. Trong thư mục **MaSV_HoVaTen_Lab08**, tạo project win32 console application : Bai3_D, với các tập tin :

- Header.h : Chứa định nghĩa dữ liệu, các khai báo nguyên mẫu các hàm hệ thống, chức năng, định nghĩa các hàm hệ thống, nhập xuất.
- ThuVien.h : Chứa các định nghĩa các hàm chức năng
- Menu.h : Chứa các hàm tổ chức menu
- Program.cpp : Chứa main() thực hiện chương trình
- Bangluong.txt : tập tin dữ liệu

Phát triển tiếp chương trình, . . .

Tham khảo phần code các tập tin nêu trên :

- a. Trong tập tin thư viện <head.h> :

```
#pragma once
```

```
//Tập tin định nghĩa du lieu va cac thao tac co ban BST, tao cay va xuat cay cac thu tu
```

```
//Kieu cua khoa la kieu chuoi, bieu din ma nhan vien
#define MAXCOT 70

typedef char KeyType[8]; //Kieu du lieu cua Khoa, la mot thanh phan cua kieu nhanvien - (ma nhan vien)

//Dinh nghia kieu thanh phan du lieu cua nut
struct nhanvien
{
    KeyType maNV; //ma nhan vien : truong khoa, dung de dua du lieu len cay
    char hoTen[25];
    int namSinh;
    char diachi[15];
    double luong;
};

//Kieu du lieu cua cac nut
struct BSNode
{
    nhanvien info;
    BSNode *left;
    BSNode *right;
};

//Kieu cay nhi phan timkiem
typedef BSNode *BSTree;

//=====
//khai bao nguyen mau
BSNode *CreateNode(nhanvien x);
void CreateBST(BSTree &root);
int InsertNode(BSTree &root, nhanvien x);
void PosOrder(BSTree root);
void InOrder(BSTree root);
void PreOrder(BSTree root);

//=====
//Tao nut voi x cho truoac
BSNode *CreateNode(nhanvien x)
{
    BSNode *p = new BSNode;
    if (p != NULL)
    {
        p->info = x;
        p->left = NULL;
        p->right = NULL;
    }
    return p;
}

//-----
//Khoi tao cay BST rong
void CreateBST(BSTree &root)
{
    root = NULL;
}

//-----
//Them x vao cay BST
int InsertNode(BSTree &root, nhanvien x)
```

```
{
//Cay khac rong
if (root != NULL)
{
    if (strcmp(root->info.maNV, x.maNV) == 0)
        return 0; // x da co san
    if ((strcmp(root->info.maNV, x.maNV) > 0))
        return InsertNode(root->left, x);
    else
        return InsertNode(root->right, x);
} //root == NULL : chen vi tri nut ngoai thich hop - la nut la sau khi chen xong

root = CreateNode(x);
if (root == NULL)
    return -1;
return 1; //thanh cong
}
//-----
//Tao cay BST tu file
int File_BST(BSTree &root, char *filename)
{
    ifstream in(filename);
    if (!in)
        return 0;

    KeyType maNV; //truong khoa, dung de dua du lieu len cay
    char hoTen[25];
    int namSinh;
    char diachi[15];
    double luong;

    int kq;
    CreateBST(root);
    nhanvien x;

    in >> maNV; strcpy_s(x.maNV, maNV);
    in >> hoTen; strcpy_s(x.hoTen, hoTen);
    in >> namSinh; x.namSinh = namSinh;
    in >> diachi; strcpy_s(x.diachi, diachi);
    in >> luong; x.luong = luong;

    kq = InsertNode(root, x);
    if (kq == 0 || kq == -1)
        return 0;
    while (!in.eof())
    {
        in >> maNV; strcpy_s(x.maNV, maNV);
        in >> hoTen; strcpy_s(x.hoTen, hoTen);
        in >> namSinh; x.namSinh = namSinh;
        in >> diachi; strcpy_s(x.diachi, diachi);
        in >> luong; x.luong = luong;

        kq = InsertNode(root, x);
        if (kq == 0 || kq == -1)
            return 0;
    }
    in.close();
    return 1;
}
```

```

}

//Xuat tieu de
void TieuDe()
{
    int i;
    cout << "\n:";
    for (i = 1; i <= MAXCOT; i++)
        cout << "=";
    cout << "\n:";
    cout << setiosflags(ios::left);
    cout << ':'
        << setw(8) << "Ma NV"
        << ':'
        << setw(25) << "Ho va Ten"
        << ':'
        << setw(5) << "NS"
        << ':'
        << setw(15) << "Dia Chi"
        << ':'
        << setw(13) << "Luong"
        << ':';
    cout << "\n:";
    for (i = 1; i <= MAXCOT; i++)
        cout << "=";
    cout << ":";
}

//Xuat 1 nhan vien
void Xuat_NV(nhanvien p)
{
    cout << endl;
    cout << setiosflags(ios::left)
        << ':'
        << setw(8) << p.maNV
        << ':'
        << setw(25) << p.hoTen
        << ':'
        << setw(5) << p.namSinh
        << ':'
        << setw(15) << p.diachi
        << ':'
        << setw(13) << setprecision(2) << setiosflags(ios::fixed) << p.luong
        << ':';
}

//Xuat cay theo thu tu truoc : NLR
void PreOrder(BSTree root)
{
    if (root != NULL)
    {
        Xuat_NV(root->info);
        PreOrder(root->left);
        PreOrder(root->right);
    }
}

//Xuat cay theo thu tu giua : LNR

```



```
void InOrder(BSTree root)
{
    if (root != NULL)
    {
        InOrder(root->left);
        Xuat_NV(root->info);
        InOrder(root->right);
    }
}
```

//Xuat cay theo thu tu sau : LRN

```
void PosOrder(BSTree root)
{
    if (root != NULL)
    {
        PosOrder(root->left);
        PosOrder(root->right);
        Xuat_NV(root->info);
    }
}
//-----
```

b. Trong tập tin thư viện <ThuVien.h> :

//Dem so nut

```
int DemSoNut(BSTree root)
{
    if (root == NULL)
        return 0;
    return 1 + DemSoNut(root->left) + DemSoNut(root->right);
}
```

//Tim nhan vien khi biet ma nhan vien

```
BSTree Search(BSTree root, KeyType x)
{
    if (root != NULL)
    {
        if (strcmp(root->info.maNV, x) == 0) //Tim thay x
            return root;
        else
            if (strcmp(root->info.maNV, x) < 0)
                return Search(root->right, x); //tim x trong cay con phai
            else
                return Search(root->left, x); //tim x trong cay con trai
    }
    return NULL; //khong co
}
```

int TinhMax(int a, int b)

```
{
    if (a >= b)
        return a;
    return b;
}
```

```
//Chiều cao của cây
int TinhChieuCao(BSTree root)
{
    int h;
    if (root == NULL)
        h = -1;
    else
        if (root->left == NULL && root->right == NULL)
            h = 0;
        else
            h = 1 + TinhMax(TinhChieuCao(root->left), TinhChieuCao(root->right));
    return h;
}

//Hủy nút có giá trị nhỏ nhất của cây con phải của root
//Input root
//Output : root (đã xóa được nút có giá trị nhỏ nhất của cây con trái root
//      x; khóa của nút bị xóa

nhavien DeleteMin(BSTree &root)
{
    nhavien k;
    if (root->left == NULL)
    {
        k = root->info;
        root = root->right;
        return k;
    }
    else
        return DeleteMin(root->left);
}

//Hủy một nút có khóa cho trước ra khỏi cây : nút thế mang là phần tử nhỏ nhất của cây con phải
//Input : x, root
//Output : 1, root (cây BST kết quả của root) nếu thành công
//      0; không thành công

int DeleteNode(nhavien x, BSTree &root)
{
    if (root != NULL)
    {
        if (strcmp(x.maNV, root->info.maNV) < 0)
            return DeleteNode(x, root->left);
        else
            if (strcmp(x.maNV, root->info.maNV) > 0)
                return DeleteNode(x, root->right);
            else //x == root->key
                if ((root->left == NULL) && (root->right == NULL)) //không có con trái, không có con
phải
                    root = NULL;
                else
                    if (root->left == NULL) //có 1 con : con phải, không có con trái
                        root = root->right;
                    else
                        if (root->right == NULL) //có 1 con : con trái, không có con phải
                            root = root->left;
                        else //có cả 2 con trái, phải
    
```

root->info = DeleteMin(root->right); //Hủy nút có giá trị nhỏ

nhat của cây con phải x

return 1; //Thành công

}

return 0;

}

int TimMuc_x(BSTree root, KeyType x)

{

int muc;

muc = 0;

BSTree T = root;

while (T != NULL)

{

if (strcmp(T->info.maNV, x) == 0)

return muc;

else

{

muc++;

if (strcmp(T->info.maNV, x) > 0)

T = T->left;

else

T = T->right;

}

return -1; //T rỗng, hoặc không có nút chứa x

}

//Tìm nhân viên biết lương theo thứ tự đầu

BSTree NLR(BSTree root, double x)

{

if (root != NULL)

{

if (root->info.luong == x)

return root;

return NLR(root->left, x);

return NLR(root->right, x);

}

}

c. Trong tập tin thư viện <Menu.h> :

void XuatMenu()

{

cout << "\n=====He thong menu=====";

cout << "\n0. Thoat khỏi chương trình";

cout << "\n1. Tạo bảng lương";

cout << "\n2. Xuất bảng lương theo thứ tự trước, giữa, cuối";

cout << "\n3. Xuất thông tin nhân viên khi biết mã nhân viên";

cout << "\n4. Thay đổi nam sinh của 1 nhân viên";

cout << "\n5. Tính chiều cao của cây";

cout << "\n6. Tính số nút của cây";

cout << "\n7. Hủy các nhân viên có lương cao nhất";

cout << "\n8. Tính mức của nút nhân viên khi biết mã nhân viên";

}

int ChonMenu(int soMenu)

{

int stt;

for (;;)

{

```

        system("CLS");
        XuatMenu();
        cout << "\nNhập 1 số trong khoảng [0,...," << soMenu << "]" để chọn menu, stt = ";
        cin >> stt;
        if (0 <= stt && stt <= soMenu)
            break;
    }
    return stt;
}
void XuLyMenu(int menu, BSTree &root)
{
    char *filename;
    int kq;
    KeyType x;
    BSTree p;
    //double luong;
    int ns;
    //=====
    switch (menu)
    {
    case 0:
        system("CLS");
        cout << "\0. Thoát khỏi chương trình\n";
        break;
    case 1:
        system("CLS");
        cout << "\n1. Tạo cây BST";
        filename = new char[50];
        do
        {
            cout << "\nfilename = ";
            cin >> filename;
            kq = File_BST(root, filename);
        } while (kq == 0);
        cout << "\nĐã chuyển dữ liệu tập tin " << filename << " vào cây BST thành công.";
        delete[] filename;
        break;
    case 2:
        system("CLS");
        cout << "\n2. Xuất cây BST theo thứ tự giữa";
        cout << "\n\nCây BST hiện hành, xuất theo thứ tự trước (NLR) :\n";
        TieuDe();
        PreOrder(root);
        cout << "\n:";
        for (int i = 1; i <= MAXCOT; i++)
            cout << "=";
        cout << ":";
        system("PAUSE");
        cout << "\n\nCây BST hiện hành, xuất theo thứ tự giữa (LNR) :\n";
        TieuDe();
        InOrder(root);
        cout << "\n:";
        for (int i = 1; i <= MAXCOT; i++)
            cout << "=";
        cout << ":";
        system("PAUSE");
        cout << "\n\nCây BST hiện hành, xuất theo thứ tự cuối (LRN) :\n";
        TieuDe();
        PosOrder(root);
    }
}

```

```
cout << "\n:";
for (int i = 1; i <= MAXCOT; i++)
    cout << "=";
cout << ":";
cout << "\nDanh sach co " << DemSoNut(root) << " nhan vien ";
system("PAUSE");
break;
```

case 3:

```
system("CLS");
cout << "\n3. Xuất thông tin nhân viên khi biết mã nhân viên";
cout << "\n\nCây BST hiện hành, xuất theo thứ tự giữa (LNR) :\n";
TieuDe();
InOrder(root);
cout << "\n:";
for (int i = 1; i <= MAXCOT; i++)
    cout << "=";
cout << ":";
cout << "\nDanh sach co " << DemSoNut(root) << " nhan vien ";
cout << "\nNhập mã nhân viên (không có ký tự trống, có phân biệt ký tự thường, HOA):\nmaNV = ";
cin >> x;
p = Search(root, x);
if (p == NULL)
    cout << "\nkhông có nhân viên nào có mã " << x;
else
{
    cout << "\nThông tin nhân viên có mã " << x << " :\n";
    TieuDe();
    Xuat_NV(p->info);
    cout << "\n:";
    for (int i = 1; i <= MAXCOT; i++)
        cout << "=";
    cout << ":";
}
break;
```

case 4:

```
system("CLS");
cout << "\n4. Thay đổi nam sinh của 1 nhân viên";
cout << "\n\nCây BST hiện hành, xuất theo thứ tự giữa (LNR) :\n";
TieuDe();
InOrder(root);
cout << "\n:";
for (int i = 1; i <= MAXCOT; i++)
    cout << "=";
cout << ":";
cout << "\nDanh sach co " << DemSoNut(root) << " nhan vien ";

cout << "\nNhập mã nhân viên (không có ký tự trống, có phân biệt ký tự thường, HOA):\nmaNV = ";
cin >> x;
p = Search(root, x);
if (p == NULL)
    cout << "\nkhông có nhân viên nào có mã " << x;
else
{
    cout << "\nThông tin nhân viên có mã " << x << " :\n";
    TieuDe();
    Xuat_NV(p->info);
    cout << "\n:";
    for (int i = 1; i <= MAXCOT; i++)
        cout << "=";
}
```

```

        cout << ":";
        cout << "\n\nNhap nam sinh moi cua nhan vien, ns = ";
        cin >> ns;
        p->info.namSinh = ns;
        cout << "\nThong tin nhan vien co ma " << x << " :\n";
        TieuDe();
        Xuat_NV(p->info);
        cout << "\n:";
        for (int i = 1; i <= MAXCOT; i++)
            cout << "=";
        cout << ":";
    }
    cout << endl;
    break;
case 5:
    system("CLS");
    cout << "\n5. Chieu cao cua cay";
    TieuDe();
    InOrder(root);
    cout << "\n:";
    for (int i = 1; i <= MAXCOT; i++)
        cout << "=";
    cout << ":";
    cout << "\nDanh sach co " << DemSoNut(root) << " nhan vien ";
    cout << "\nChieu cao cua cay BST : h = " << TinhChieuCao(root);
    cout << endl;
    break;
case 6:
    system("CLS");
    cout << "\n5. So nut cua cay";
    TieuDe();
    InOrder(root);
    cout << "\n:";
    for (int i = 1; i <= MAXCOT; i++)
        cout << "=";
    cout << ":";
    cout << "\nDanh sach co " << DemSoNut(root) << " nhan vien ";
    cout << endl;
    break;
case 7:
    system("CLS");
    cout << "\n7. Tinh muc cua nut nhan vien khi biet ma nhan vien";
    cout << endl;
    cout << "\n\nCay BST hien hanh, xuất theo thu tu giữa (LNR) :\n";
    TieuDe();
    InOrder(root);
    cout << "\n:";
    for (int i = 1; i <= MAXCOT; i++)
        cout << "=";
    cout << ":";
    cout << "\nDanh sach co " << DemSoNut(root) << " nhan vien ";

    cout << "\n\nNhap ma nhan vien(khong co KT trang, co phan biet KT thương, HOA), x = ";
    cin >> x;
    p = Search(root, x);
    if (p == NULL)
        cout << "\nkhong co nhan vien nao co ma " << x;
    else
        cout << "\nMuc cua nhan vien co ma " << x << " la : " << TimMuc_x(root, x);

```

```

        break;

    case 8:
        system("CLS");
        cout << "\n8. Huy các nhân viên có lương cao nhất";
        cout << "\n\nCây BST hiện hành, xuất theo thứ tự giữa (LNR) :\n";
        Tieude();
        InOrder(root);
        cout << "\n:";
        for (int i = 1; i <= MAXCOT; i++)
            cout << "=";
        cout << ":";
        cout << "\nDanh sách có " << DemSoNut(root) << " nhân viên ";

        cout << endl;
        break;
    }
}

```

d. Trong tập tin <Program.cpp>

```

#include <iostream>
#include <fstream>
#include <string.h>
#include <iomanip>
using namespace std;
#include "Header.h"
#include "ThuVien.h"
#include "Menu.h"
void ChayChuongTrinh();
int main()
{
    ChayChuongTrinh();
    return 1;
}
void ChayChuongTrinh()
{
    BSTree root = NULL;
    int menu, soMenu = 7;
    do
    {
        menu = ChonMenu(soMenu);
        XuLyMenu(menu, root);
        system("PAUSE");
    } while (menu > 0);
}

```

E. Bài tập.

Bài 1.

Viết chương trình tùy chọn thực hiện các thao tác cơ bản trên **cây nhị phân**, khóa các nút trong cây là số nguyên :

1. Tạo cây nhị phân cân đối (tại mỗi nút, số con trái và số con phải lệch nhau không quá 1)
2. Xuất các phần tử trên cây theo thứ tự đầu (NLR), giữa (LNR), cuối (LRN).
3. Đếm số nút của cây
4. Xác định chiều cao của cây
5. Đếm số nút của cây
6. Đếm số nút của cây ở mức K
7. Đếm số nút lá của cây
8. Đếm số nút có đúng hai nút con khác rỗng
9. Huy nút có khóa cho trước
10. Chèn một khóa vào cây

11. Duyệt cây theo chiều rộng (BFS)
12. Duyệt cây theo chiều sâu (DFS)

Dữ liệu được cho trong các tập tin :

- “Test.txt” :

25 37 10 18 29 50 3 1 6 5 12 20 35 13 32 41

Bài 2.

Bảng điểm của các học viên của một trung tâm đào tạo chứa các thông tin của học viên như sau :

- Mã học viên, //chuỗi có đúng 7 ký tự, không có ký tự trắng
- Họ và Tên của học viên, // chuỗi có không quá 25 ký tự
- Năm sinh, số nguyên dương có 4 ký số
- Địa chỉ, // chuỗi có không quá 15 ký tự
- Điểm //số thực từ 0 đến 10

Viết chương trình tùy chọn thực hiện trên bảng điểm học viên, với bảng điểm được cài đặt bằng cây nhị phân tìm kiếm với khóa là mã học viên :

0. Thoát khỏi chương trình
 1. Tạo bảng điểm học viên (chuyển dữ liệu từ tập tin cho trước vào cây nhị phân tìm kiếm dựa vào khóa mã học viên.
 2. Xem bảng điểm học viên theo thứ tự trước (NLR), thứ tự giữa (LNR), thứ tự cuối (LRN)
 3. Thay đổi địa chỉ của học viên khi biết mã học viên
 4. Hủy học viên có mã nhân viên cho trước ra khỏi bảng điểm .
 5. Hủy các học viên có điểm < 5.
- Tập tin dữ liệu “*bd.txt*” lưu trữ thông tin các học viên trong bảng điểm với định dạng như sau :
 - Tập tin có 12 hàng, mỗi hàng gồm 5 cột (các cột tách biệt bằng ký tự tách như ký tự trắng, tab,...) chứa các thông tin một học viên : Cột 1 chứa mã học viên, cột 2 chứa Họ và Tên học viên, cột 3 chứa năm sinh (NS) học viên, cột 4 chứa địa chỉ học viên, cột cuối cùng chứa điểm của học viên.
 - Các mã học viên đôi một khác nhau. Mã học viên xác định duy nhất 1 học viên.
 - Các chuỗi lưu trữ thông tin về họ và tên học viên, địa chỉ của học viên có thể gồm nhiều từ, các từ được nối với nhau bởi dấu gạch dưới.

Mã HV	Họ và Tên học viên	NS	Địa chỉ	Điểm
1612045	Nguyen_Tuan_Vo	1980	Lam_Dong	9
1513452	Tran_Van_Ngoc	1974	Khanh_Hoa	6
1615332	Le_Thi_Hoa	1974	Binh_Dinh	8
1522145	Le_Minh	1986	Khanh_Hoa	4
1614432	Nguyen_Vo	1985	Phu_Yen	4
1324042	Vo_Ngoc_Hoa	1983	Ha_Noi	4
1522032	Van_Thi_Hanh	1984	Lam_Dong	9
1422052	Vo_Ngoc_Thai	1985	Lam_Dong	9
1513210	Ly_Van_Hoa	1985	Ninh_Thuan	8
1623045	Tran_Le_Trong	1991	Ha_Noi	4
1322140	Tran_Minh_Vuong	1990	Binh_Dinh	9
1430432	Nguyen_Vo	1975	Lam_Dong	5

Bài 3.

Viết chương trình thực hiện các chức năng sau:

- a. Nhập vào một biểu thức số học đơn giản. Biểu diễn biểu thức số học lên cây nhị phân. Kiểm tra cú pháp của biểu thức.
- b. Xuất ra biểu thức số học đó dưới dạng: tiền tố, trung tố, hậu tố.
- c. Tính giá trị của biểu thức.

Bài 4. Kiểu dữ liệu trừu tượng: Từ điển

Cài đặt KDLTT từ điển Anh-Việt bằng cây tìm kiếm nhị phân theo mô tả như sau: Một từ điển có nhiều từ, một từ có thể có nhiều nghĩa. Với mỗi từ, ta cần lưu từ gốc tiếng Anh (key) và nghĩa tiếng Việt của từ đó (meaning).

LAB 09. Cân bằng (AVL)

A. Mục tiêu

- Cài đặt cây AVL

B. Yêu cầu

- Buổi thực tập thứ 10 : làm bài 1D

C. Ôn tập

1. Cây cân bằng và chỉ số cân bằng

Cây tìm kiếm nhị phân cân bằng là cây tìm kiếm nhị phân mà tại mỗi nút của nó, độ cao của cây con trái và độ cao của cây con phải chênh lệch nhau không quá 1.

Chỉ số cân bằng (Balance factor) của một nút p là hiệu của chiều cao cây con phải và chiều cao cây con trái của nó. Nghĩa là:

$$\text{BalFactor}(p) = \text{height}(p.\text{RightChild}) - \text{height}(p.\text{LeftChild})$$

Việc thêm hay hủy một nút trên cây AVL có thể làm cây tăng hay giảm chiều cao, khi đó, ta cần phải cân bằng lại cây. Để giảm tối đa chi phí cân bằng lại cây, ta chỉ cân bằng lại ở phạm vi cục bộ. Vì thế, với mỗi nút của cây, ngoài các thuộc tính thông thường như cây nhị phân, ta cần lưu trữ thêm thông tin về chỉ số cân bằng.

Để đơn giản, ta quy ước: $\text{EH} = 0$, $\text{RH} = 1$, $\text{LH} = -1$

$\text{BalFactor}(p) = \text{EH} \Leftrightarrow$ 2 cây con của p cao bằng nhau

$\text{BalFactor}(p) = \text{RH} \Leftrightarrow$ cây lệch phải hay cây con phải của p cao hơn cây con trái

$\text{BalFactor}(p) = \text{LH} \Leftrightarrow$ cây lệch phải hay cây con trái của p cao hơn cây con phải

2. Các trường hợp mất cân bằng

Xét trường hợp chèn thêm một nút mới v vào một nút u trên cây cân bằng T . Do v chỉ có thể được chèn vào đúng một trong hai cây con của u nên nhiều nhất là v có thể làm tăng chiều cao của một trong hai cây con đó. Nếu v không làm tăng chiều cao của cây con nào hoặc v làm tăng chiều cao của một cây con nhưng trước đó cây con này có chiều cao nhỏ hơn hoặc bằng chiều cao của cây con kia thì tính cân bằng AVL tại đỉnh u vẫn giữ nguyên.

Tính cân bằng AVL tại u chỉ có thể bị phá vỡ khi v làm tăng chiều cao của cây con có chiều cao lớn hơn trong hai cây con của u . Cũng như vậy, nếu v không làm tăng chiều cao của chính u thì v không làm thay đổi hệ số cân bằng tại các đỉnh tiền bối của u .

Mặt khác, nút v luôn được thêm vào với tư cách là con của một nút trước đó là lá hoặc nửa lá. Nếu cha của v trước khi thêm v là nửa lá thì chiều cao của cây con gốc cha của v không thay đổi sau khi thêm v còn hệ số cân bằng tại đỉnh cha này bằng 0. Khi đó tất cả các nút tiền bối của cha của v không thay đổi hệ số cân bằng. Tính cân bằng AVL được giữ vững trên toàn bộ cây T .

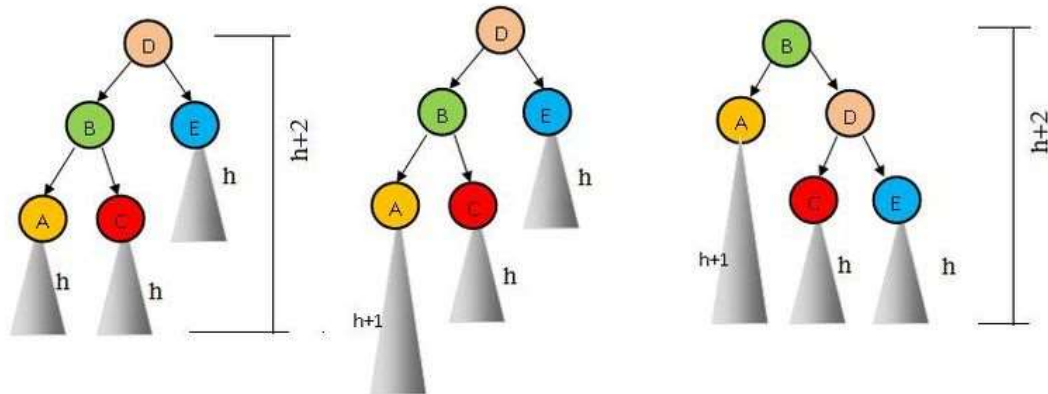
Nếu đỉnh cha của v trước khi chèn v là lá, gọi u là đỉnh tiền bối của v có mức cao nhất mà tính cân bằng AVL bị phá vỡ. Như vậy bốn trường hợp sau có thể phá vỡ tính cân bằng AVL tại u

- Trước khi chèn cây con gốc u lệch trái và v làm tăng chiều cao của cây con trái.
 - Sau khi chèn cây con trái lệch trái (Case LL)
 - Sau khi chèn cây con trái lệch phải (Case LR)
- Trước khi chèn cây con gốc u lệch phải và v làm tăng chiều cao của cây con phải.
 - Sau khi chèn cây con phải lệch phải (Case RR)
 - Sau khi chèn cây con phải lệch trái. (Case RL)

Xét tương tự cho trường hợp xóa một nút khỏi cây. Lưu ý, việc cân bằng lại có thể xảy ra theo phản ứng dây chuyền.

3. Cân bằng lại cây

Trường hợp cây con trái lệch trái: Sử dụng phép quay đơn Left-Left (RotateLL)

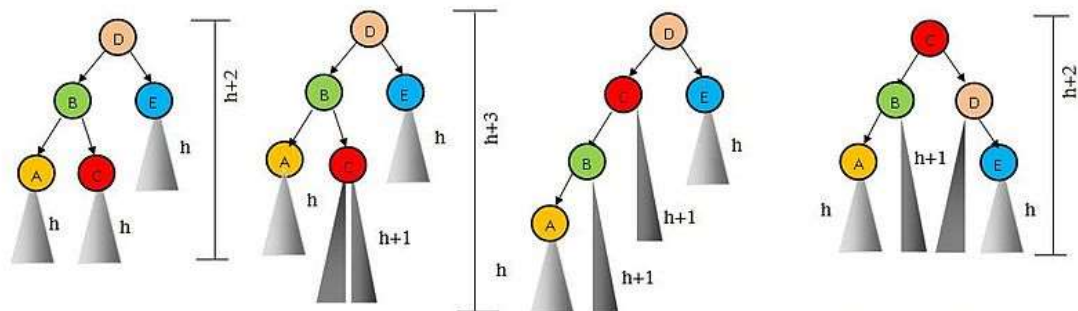


1. Cây lệch trái trước khi chèn. Chiều cao cây là $h+2$,

2. Chèn một phần tử vào cây con trái của cây con trái làm cho cây mất cân bằng

3. Sau phép quay phải cây trở thành cân bằng, chiều cao cây vẫn là $h+2$

Trường hợp cây con trái lệch phải: Sử dụng phép quay Left-Right (RotateLR)



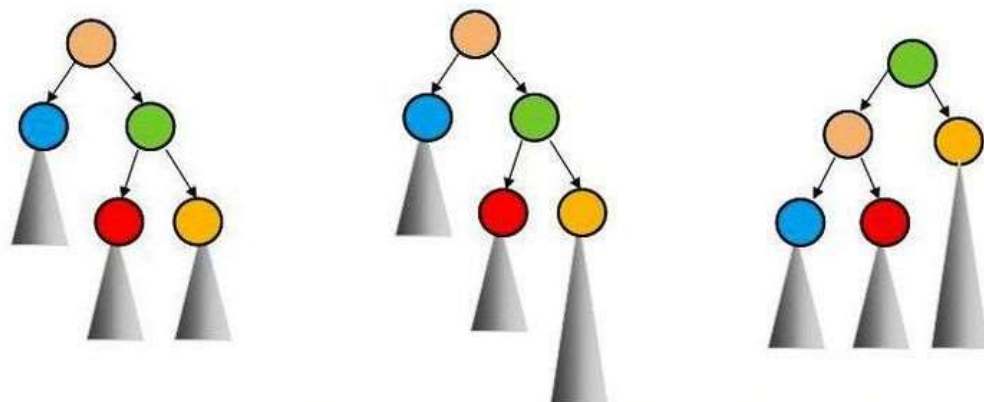
1. Cây ban đầu lệch trái, cây con trái cân bằng. Chiều cao cây là $h+2$

2. Phép chèn vào cây con trái làm cho cây con trái lệch phải.

3. Phép xoay trái cây con trái đưa cây con trái thành cây lệch trái

4. Sau phép xoay phải cây, cây trở thành cân bằng AVL. Chiều cao vẫn là $h+2$

Trường hợp cây con phải lệch phải: Sử dụng phép quay Right-Right (RotateRR)

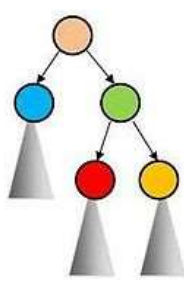


1. Cây lệch phải

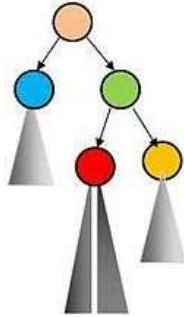
2. Sau khi chèn, cây con phải lệch phải

3. Sau phép quay trái cây trở thành cân bằng AVL

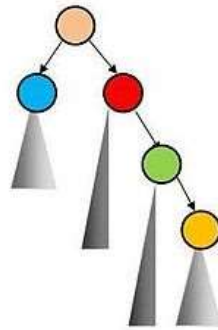
Trường hợp cây lệch phải, cây con phải lệch trái: Sử dụng phép quay Right-Left (RotateRL)



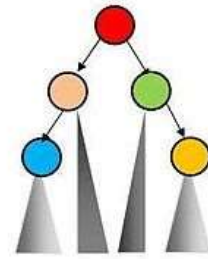
1. Cây AVL lệch phải



2. Sau phép chèn vào cây con phải, cây con phải lệch trái.



3. Sau phép quay phải cây con phải. Cây con phải trở thành lệch phải



4. Sau phép quay trái, cây trở thành cân bằng AVL.

D. Hướng dẫn

Bài 1.

Cài đặt các thao tác trên cây tìm kiếm nhị phân cân bằng (AVL)

Viết hàm trong tập tin avltree.h

a. Tìm nút trên cây chứa giá trị key cho trước (dạng đệ quy)

Search(root, key, &parent) ≡

```
{
    Nếu (root = null) thì trả về null;           // Không tìm thấy
    Nếu (root.Key = key) thì trả về root;        // Tìm thấy
    Ngược lại
    {
        Gán parent = root;                      // Cập nhật nút cha
        Nếu (root.Key > key) thì
            Trả về kết quả tìm kiếm key trong cây con trái của root.
        Ngược lại
            Trả về kết quả tìm kiếm key trong cây con phải của root.
    }
}
```

b. Duyệt cây theo thứ tự NLR (PreOrder), LNR (Inorder), LRN (PosOrder) dạng đệ quy

c. Phép quay đơn Left-Left. Sử dụng khi cây con bên trái lệch trái hoặc cân bằng.

RotateLL(&T) ≡

```
{
    Gán T1 = Cây con trái của T;
    Gán Cây con trái của T = Cây con phải của T1;
    Gán Cây con phải của T1 = T;
    Nếu (Cây con trái T1 đang cân bằng) thì
    {
        Gán hệ số cân bằng của T = LH;           // Sau khi quay
        Gán hệ số cân bằng của T1 = RH;           // T lệch trái
                                                    // T1 lệch phải
    }
    Ngược lại, nếu (Cây con trái T1 đang lệch trái) thì
    {
        Gán hệ số cân bằng của T = EH;           // T cân bằng
        Gán hệ số cân bằng của T1 = EH;           // T1 cân bằng
    }
    T = T1;                                       // Thay thế nút T bởi nút T1 (đổi vai trò của T1)
}
```

d. Phép quay đơn Right-Right. Sử dụng khi cây con bên phải lệch phải hoặc cân bằng.

RotateRR(&T) ≡

```
{
    Gán T1 = Cây con phải của T;
    Gán Cây con phải của T = Cây con trái của T1;
```

```
Gán Cây con trái của T1 = T;
Nếu (Cây con phải T1 đang cân bằng) thì
{
    Gán hệ số cân bằng của T = RH;           // Sau khi quay
    Gán hệ số cân bằng của T1 = LH;           // T lệch phải
                                           // T1 lệch trái
}
Ngược lại, nếu (Cây con phải T1 đang lệch phải) thì
{
    Gán hệ số cân bằng của T = EH;           // T cân bằng
    Gán hệ số cân bằng của T1 = EH;           // T1 cân bằng
}
T = T1;                                     // Thay thế nút T bởi nút T1 (đổi vai trò của T1)
}
```

- e. Phép quay kép Left-Right. Sử dụng khi cây con bên trái lệch phải.

```
RotateLR(T) ≡
{
    Gán T1 = Cây con trái của T;
    Gán T2 = Cây con phải của T1;
    Gán Cây con phải của T1 = Cây con trái của T2;
    Gán Cây con trái của T = Cây con phải của T2;
    Gán Cây con trái của T2 = T1;
    Gán Cây con phải của T2 = T;
    Nếu (Cây con T2 đang lệch trái) thì
    {
        Gán hệ số cân bằng của T = RH;           // Sau khi quay
        Gán hệ số cân bằng của T1 = EH;           // T lệch phải
                                           // T1 cân bằng
    }
    Ngược lại, Nếu (Cây con T2 đang cân bằng) thì
    {
        Gán hệ số cân bằng của T = EH;           // Sau khi quay
        Gán hệ số cân bằng của T1 = EH;           // T cân bằng
                                           // T1 cân bằng
    }
    Ngược lại, nếu (Cây con T1 đang lệch phải) thì
    {
        Gán hệ số cân bằng của T = EH;           // T cân bằng
        Gán hệ số cân bằng của T1 = LH;           // T1 lệch trái
    }
    Gán hệ số cân bằng của T2 = EH;           // T2 cân bằng
    T = T2;                                     // Thay thế nút T bởi nút T2 (đổi vai trò của T2)
}
```

- f. Phép quay kép Right-Left. Sử dụng khi cây con bên phải lệch trái.

```
RotateRL(&T) ≡
{
    Gán T1 = Cây con phải của T;
    Gán T2 = Cây con trái của T1;
    Gán Cây con trái của T1 = Cây con phải của T2;
    Gán Cây con phải của T = Cây con trái của T2;
    Gán Cây con trái của T2 = T;
    Gán Cây con phải của T2 = T1;
    Nếu (Cây con T2 đang lệch trái) thì
    {
        Gán hệ số cân bằng của T = EH;           // Sau khi quay
        Gán hệ số cân bằng của T1 = RH;           // T cân bằng
                                           // T1 lệch phải
    }
    Ngược lại, Nếu (Cây con T2 đang cân bằng) thì
    {
        Gán hệ số cân bằng của T = EH;           // Sau khi quay
        Gán hệ số cân bằng của T1 = EH;           // T cân bằng
                                           // T1 cân bằng
    }
}
```

```

    }
    Ngược lại, nếu (Cây con T1 đang lệch phải) thì
    {
        Gán hệ số cân bằng của T = LH;           // T lệch trái
        Gán hệ số cân bằng của T1 = EH;           // T1 cân bằng
    }
    Gán hệ số cân bằng của T2 = EH;           // T2 cân bằng
    T = T2;           // Thay thế nút T bởi nút T2 (đổi vai trò của T2)
}

g. Cân bằng lại cây khi cây lệch trái
ReBalanceLeft(&T) =
{
    Gán T1 = Cây con bên trái của T;
    switch (Hệ số cân bằng của T)
    {
        LH: Quay đơn Left-Left tại nút T; Trả về BAL;
        EH: Quay đơn Left-Left tại nút T; Trả về DEV;
        RH: Quay kép Left-Right tại nút T; Trả về BAL;
        Default: Trả về BAL;
    }
}

h. Cân bằng lại cây khi cây lệch phải
ReBalanceRight(&T) =
{
    Gán T1 = Cây con bên phải của T;
    switch (Hệ số cân bằng của T)
    {
        LH: Quay kép Right-Left tại nút T; Trả về BAL;
        EH: Quay đơn Right-Right tại nút T; Trả về DEV;
        RH: Quay kép Right-Right tại nút T; Trả về BAL;
        Default: Trả về BAL;
    }
}

i. Thêm một phần tử mới vào cây
Insert(&T, item) =
{
    Nếu (Cây T = rỗng) thì
    {
        Tạo nút chứa dữ liệu item và gán cho T;
        Nếu (Cây T = rỗng) thì trả về FAIL;           // Không thể chèn
        Ngược lại, trả về BAL;
    }
    Ngược lại
    {
        Gán result = ZERO;
        Nếu (T.Key = item) thì trả về FAIL;           // phần tử đã có
        Ngược lại, nếu (T.Key > item) thì
        {
            Gán result = Insert(T.LeftChild, item);
            Nếu (result khác BAL) thì trả về result;
            Ngược lại
            {
                switch (Hệ số cân bằng của T)
                {
                    LH: Cân bằng lại vì T lệch trái; Trả về DEV;
                    EH: Gán hệ số cân bằng của T = LH; Trả về BAL;
                    RH: Gán hệ số cân bằng của T = EH; Trả về DEV;
                }
            }
        }
    }
}

```

```

    }
}
Ngược lại
{
    Gán result = Insert(T.RightChild, item);
    Nếu (result khác BAL) thì trả về result;
    Ngược lại
    {
        switch (Hệ số cân bằng của T)
        {
            LH: Gán hệ số cân bằng của T = EH; Trả về DEV;
            EH: Gán hệ số cân bằng của T = RH; Trả về BAL;
            RH: Cân bằng lại vì T lệch phải; Trả về DEV;
        }
    }
}
}

j. Tìm nút trái nhất của cây con bên phải Q để thay thế cho P
SearchStandFor(&p, &q) ≡
{
    Nếu (q có cây con bên trái) thì
    {
        Gán kq = Tìm nút trong cây con bên trái Q để thay thế cho P;
        Nếu (kq < 2) thì trả về kq; // Không thể chèn
        switch (Hệ số cân bằng của q)
        {
            LH: Gán hệ số cân bằng của q = EH; Trả về BAL; // 2
            EH: Gán hệ số cân bằng của q = RH; Trả về DEV; // 1
            RH: Trả về kết quả sau khi Cân bằng lại cây tại nút q;
        }
    }
    Ngược lại
    {
        Gán dữ liệu từ nút q vào nút p;
        Gán p = q; // Đổi vai trò của p, q
        Gán q = Cây con trái của q;
        return BAL;
    }
}

k. Xóa một nút ra khỏi cây
iDelete(&T, item) ≡
{
    Nếu (Cây T = rỗng) thì trả về FAIL; // Không thể xóa
    Khởi tạo biến result = ZERO;
    Nếu (item nằm trong cây con trái của T) thì
    {
        Gán result = Delete(T.LeftChild, item);
        Nếu (result != BAL) thì trả về result;
        switch (Hệ số cân bằng của T)
        {
            LH: Gán hệ số cân bằng của T = EH; Trả về BAL;
            EH: Gán hệ số cân bằng của T = RH; Trả về DEV;
            RH: Trả về kết quả sau khi Cân bằng lại vì T lệch phải;
        }
    }
}

```

```

Ngược lại, Nếu (item nằm trong cây con phải của T) thì
{
    Gán result = Delete(T.RightChild, item);
    Nếu (result != BAL) thì trả về result;
    switch (Hệ số cân bằng của T)
    {
        LH: Trả về kết quả sau khi Cân bằng lại vì T lệch trái;
        EH: Gán hệ số cân bằng của T = LH; Trả về DEV;
        RH: Gán hệ số cân bằng của T = EH; Trả về BAL;
    }
}
Ngược lại // TRường hợp tìm thấy nút p chứa item
{
    Gán p = T;
    Nếu (T không có con trái) thì
    {
        Gán T = Cây con phải của T;
        Gán result = BAL;
    }
    Ngược lại, Nếu (T không có con phải) thì
    {
        Gán T = Cây con trái của T;
        Gán result = BAL;
    }
    Ngược lại
    {
        Gán result = SearchStandFor(p, p.RightChild);
        Nếu (result != BAL) thì trả về result;
        switch (Hệ số cân bằng của T)
        {
            LH: Trả về kết quả sau khi Cân bằng lại vì T lệch trái;
            EH: Gán hệ số cân bằng của T = LH; Trả về BAL;
            RH: Gán hệ số cân bằng của T = EH; Trả về BAL;
        }
    }
}
Trả về result;
}

```

1. Xây dựng cây AVL từ một tập dữ liệu cho trước
 BuildTree(&T, items[],count) =
 {
 Tạo cây rỗng T;
 for (int i=0; i<count; i++)
 {
 Gọi hàm chèn phần tử items[i] vào cây T;
 }
 }

m. Viết hàm xuất hệ số cân bằng và dữ liệu chứa trong các nút của cây theo thứ tự NLR

Bài 2. Viết chương trình thực hiện các thao tác trên cây AVL với dữ liệu có kiểu chuỗi và được lấy từ tập tin. Ngoài ra, bổ sung vào cấu trúc của nút một con trỏ trỏ đến nút cha của nó.

Bài 3. Kiểu dữ liệu trừu tượng: Từ điển

Cài đặt KDLTT từ điển Anh-Việt bằng cây tìm kiếm nhị phân theo mô tả như sau: Một từ điển có nhiều từ, một từ có thể có nhiều nghĩa. Với mỗi từ, ta cần lưu từ gốc tiếng Anh (key) và nghĩa tiếng Việt của từ đó (meaning). Sau đây là một số gợi ý để cài đặt.

ÔN TẬP CUỐI KỲ

TRƯỜNG