

Trường Đại Học Công Nghệ Sài Gòn
Khoa Công Nghệ Thông Tin
---oOo---



Giáo trình thực hành:

KỸ THUẬT LẬP TRÌNH

(Lưu hành nội bộ)

Năm 2020

I. MỤC TIÊU:

Sau khi thực hành xong bài này, sinh viên được ôn tập :

- Ôn tập cấu trúc lệnh rẽ nhánh.
- Ôn tập cấu trúc lệnh lặp.
- Ôn tập hàm.

II. TÓM TẮT LÝ THUYẾT:

1. KHAI BÁO VÀ ĐỊNH NGHĨA HÀM:

- Khai báo hàm trong file **thuvien.h**

<Kiểu dữ liệu> <tên hàm>([danh sách tham số hình thức]);

Ví dụ:

```
Lab1_thuvien.h*  ↗ ✕  
#include "iostream"  
using namespace std;  
  
int tinhDTB(int toan, int ly, int hoa);
```

- Định nghĩa hàm trong file **caidat.cpp**

<Kiểu dữ liệu> <tên hàm>([danh sách tham số hình thức])

```
{  
  
    //Các lệnh trong thân hàm  
  
    [return gia_tri;]  
  
}
```

Ví dụ:

```
Lab1_caidat.cpp*  ↗ ✕  
#include "Lab1_thuvien.h"  
  
int TinhDTB(int toan, int ly, int hoa)  
{  
    int dtb;  
    dtb = (toan*3 + ly*2 + hoa) / 6;  
    return dtb;  
}
```

- Gọi hàm

<tên hàm>([danh sách tham số thực]);

Ví dụ:

```
Lab1_chuongtrinh.cpp* X
#include "Lab1_thuvien.h"

void main() {
    int toan, ly, hoa, dtb;
    cout<<"\nNhap diem toan, ly, hoa: ";
    cin>>toan>>ly>>hoa;
    dtb = tinhDTB(toan, ly, hoa);
    cout<<"\nDiem TB: "<<dtb;
    system("pause");
}
```

2. MỘT SỐ QUY TẮC VỀ HÀM

- **Hàm có giá trị trả về:** <Kiểu dữ liệu> được khai báo là tên kiểu dữ liệu của giá trị trả về, giá trị này được tính toán trong hàm và được trả về bởi lệnh **return**.
- **Hàm không có giá trị trả về:** <Kiểu dữ liệu> được khai báo là **void**.
- **Tham số hình thức và tham số thực:**
 - **Tham số hình thức:** Tham số dùng khi khai báo, và tham số này sẽ được dùng tạm trong quá trình định nghĩa hàm
 - **Tham số thực:** Là tham số được cung cấp khi hàm được sử dụng, tham số này sẽ thay thế vào vị trí của tham số hình thức hay nói cách khác là gán tham số thực tế cho tham số hình thức.
 - Ví dụ: Tham số hình thức và tham số thực.

```
void ham(int n); //Tham số hình thức void main()
{
    int a;
    ham(a); // Tham số thực
}
void ham(int n) { // Tham số hình thức
    .....
}
```

- **Cách truyền tham số:**
 - **Truyền theo tham trị:** giá trị của tham số không thay đổi sau lời gọi hàm.
 - **Truyền theo tham biến:** giá trị của tham số bị thay đổi sau lời gọi hàm (nếu hàm có lệnh làm thay đổi giá trị tham số).

3. CẤU TRÚC RẼ NHÁNH

CÚ PHÁP	VÍ DỤ
<ul style="list-style-type: none">▪ Cấu trúc if: if (biểu thức điều kiện) Khối lệnh 1; else Khối lệnh 2; ▪ Cấu trúc switch: switch (biểu thức) { case <giá trị 1>: [lệnh 1; break;] case <giá trị 2>: [lệnh 2; break;] ... case <giá trị n-1>: [lệnh n-1; break;] [default: lệnh n;] }	<pre>int a = 5, b = 7; if (a==b) cout<<"a bằng b"; else cout<<"a khác b"; int thang = 4; switch (thang) { case 1: case 2: case 3: cout<<"Qui I"; break; case 4: case 5: case 6: cout<<"Qui II"; break; case 7: case 8: case 9: cout<<"Qui III"; break; case 10: case 11:</pre>

	<pre>case 12: cout<<"Qui IV"; break; default: cout<<"Thang sai!"; }</pre>
--	---

4. CẤU TRÚC LẶP

CÚ PHÁP	VÍ DỤ
<ul style="list-style-type: none">▪ Cấu trúc for: for (biểu thức 1; biểu thức 2; biểu thức 3) Khối lệnh;▪ Cấu trúc while: while (biểu thức điều kiện) Khối lệnh;▪ Cấu trúc do...while: do { Khối lệnh; } while (biểu thức điều kiện);	<pre>int kq = 1; for (int i=1; i<=n; i++) kq = kq * i; int kq = 1, i = 1; while(i<=n) { kq = kq * i; i++; } kq = 1, i = 1; do { kq = kq * i; i++; } while(i<=n);</pre>

III. NỘI DUNG THỰC HÀNH:

1/ Chương trình mẫu: (3 điểm)

➤ Cho đoạn mã trong file **lab1_thuvien.h** như sau:

Dòng	Mã lệnh	Ghi chú
1	#include <iostream>	
2	using namespace std;	

Bài 1: ÔN TẬP CẤU TRÚC LỆNH Rẽ NHÁNH, LỆNH LẬP, HÀM

3		
4	void giaiPTB1(int a, int b, double &x);	
5	double haiMuN(int n);	

➤ Cho đoạn mã trong file **lab1_caidat.cpp** như sau:

Dòng	Mã lệnh	Ghi chú
1	#include "lab1_thuvien.h"	
2		
3	//Hàm giải phương trình bậc 1: $ax + b = 0$	
4	void giaiPTB1(int a, int b, double &x)	
5	{	
6	x = -b / a;	
7	}	
8		
9	//Hàm tính 2^n	
10	double tinh2MuN(int n)	
11	{	
12	double s = 1;	
13	for (int i = 1; i < n; i++)	
14	s *= 2;	
15	return s;	
16	}	

➤ Cho đoạn mã trong file **lab1_chuongtrinh.cpp** như sau:

Dòng	Mã lệnh	Ghi chú
1	#include "lab1_thuvien.h"	
2		
3	void main()	
4	{	
5	int a, b;	
6	double x;	
7	cout << "\nGiai phuong trinh bac 1 ($ax + b =$	
8	0):";	
9	cout << "\nNhap a=";	
10	cin >> a;	
11	cout << "\nNhap b=";	
12	cin >> b;	
13	giaiPTB1(a, b, x);	
14	cout << "\n\nNghiem cua phuong trinh la : " << x;	
15	int n;	
16	cout << "\nTinh 2 mu n:";	
17	cout << "\nNhap n=";	
18	cin >> n;	
19	cout << "\n2 mu " << n << " = " << tinh2MuN(n);	
20	system("pause");	
21	}	

Yêu cầu sinh viên:

1. Hàm `tinh2MuN(int n)` đã sai. SV hãy sửa lại để ra đúng kết quả 2^n .
2. Sinh viên thực thi chương trình với các giá trị nhập vào như sau:
 - $a = 1, b = 2, n = 2$: cho biết kết quả? Giải thích tại sao kết quả không ra số thực? SV sửa lại mã lệnh để cho kết quả nghiệm là số thực.
 - $a = 0, b = 2, n = 3$: cho biết kết quả? Giải thích tại sao chương trình báo lỗi khi thực thi? SV sửa lại mã lệnh để cho kết quả đúng như sau: nếu $a = 0$ thì xuất ra màn hình “Không chia cho số 0”, ngược lại thì tính nghiệm vào tham số x .
3. Viết lại hàm `tinh2MuN(int n)` bằng vòng lặp `while`.

2/ Chỉnh sửa số liệu theo yêu cầu: (2 điểm)

SV sửa lại hàm `tinh2MuN` để tính được cả số mũ âm, lưu ý: $2^{-n} = 1 / (2^n)$

3/ Bài tập tổng hợp: (3 điểm)

1. Viết hàm giải phương trình bậc 2 ($ax^2 + bx + c = 0$) :
`int giaiPTB2(int a, int b, int c, double &x1, double &x2)`
 - Kết quả trả về :
 - 0 : phương trình vô nghiệm.
 - 1: phương trình có 1 nghiệm.
 - 2: phương trình có 2 nghiệm.
 - 3: phương trình vô số nghiệm
 - $x1, x2$: là 2 nghiệm của phương trình.
2. Viết hàm tính x^y , x và y là 2 tham số của hàm.

4/ Bài tập làm thêm: (2 điểm)

1. Cho số nguyên dương n , viết hàm kiểm tra n có phải là lũy thừa cơ số 2 hay không?
Ví dụ: $n=16 = 2^4 \Rightarrow 16$ là lũy thừa cơ số 2.
2. Viết hàm tìm ước số chung lớn nhất của 2 số nguyên.
3. Viết hàm tìm bội số chung nhỏ nhất của 2 số nguyên.
4. Viết hàm kiểm tra số n có phải là số nguyên tố hay không?
5. Viết hàm xuất ra n số nguyên tố đầu tiên.
Ví dụ: $n = 5 \Rightarrow$ xuất ra: 2 3 5 7 11
6. Viết hàm xuất ra các số nguyên tố $\leq n$.
Ví dụ: $n = 20 \Rightarrow$ xuất ra: 2 3 5 7 11 13 17 19
7. Viết hàm kiểm tra số nguyên dương n có phải là số hoàn thiện không?
8. Viết hàm tính tổng các ký số của 1 số nguyên.
Ví dụ: số 123 có tổng các ký số $= 1+2+3 = 6$

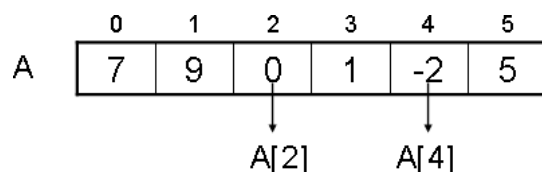
I. MỤC TIÊU:

Sau khi thực hành xong bài này, sinh viên được ôn tập :

- Ôn tập kiểu dữ liệu struct, kiểu mảng một chiều. Xây dựng kiểu dữ liệu chứa dãy số nguyên.
- Nhập/xuất dãy số nguyên trên thiết bị nhập/xuất chuẩn.
- Phép duyệt dãy số.
- Thuật toán: tính tổng, tích, đếm.

II. TÓM TẮT LÝ THUYẾT:**1. MẢNG MỘT CHIỀU:**

- Hình ảnh mảng 1 chiều:



CÚ PHÁP	VÍ DỤ
<ul style="list-style-type: none">▪ Khai báo: <Kiểu dữ liệu> <Tên mảng>[Số phần tử tối đa]; Hoặc: <Kiểu dữ liệu> <Tên mảng>[] = {ds các giá trị};▪ Truy cập đến các phần tử của mảng: <Tên mảng>[chỉ số] <p>Trong đó: $0 \leq \text{chỉ số} < \text{số phần tử tối đa}$</p>	<p>int a[50];</p> <p>int b[] = {1,2,3,4,5};</p> <p>a[2] => truy xuất đến phần tử thứ 3 của mảng, có giá trị là 0</p> <p>a[4] => truy xuất đến phần tử thứ 5 của mảng, có giá trị là -2</p>

2. KIỂU CẤU TRÚC

CÚ PHÁP	VÍ DỤ
<ul style="list-style-type: none">Khai báo: struct <tên kiểu> { thành phần 1; thành phần 2; ... thành phần n; };Khai báo biến kiểu cấu trúc: <Kiểu cấu trúc> <danh sách các biến>;Truy cập đến các thành phần kiểu cấu trúc: <biểu kiểu cấu trúc>.<tên thành phần>	Khai báo kiểu SOPHUC gồm 2 thành phần: thực và ảo struct SOPHUC { float thuc; float ao; }; SOPHUC a, b; b.ao, b.thuc, a.ao, a.thuc

3. XÂY DỰNG DÃY SỐ ARRAY

- Ý tưởng:**

Ta thấy đối với mảng một chiều, muốn duyệt mảng phải biết được số phần tử, vì vậy, để cho đơn giản, ta tạo ra kiểu dữ liệu DAYSO có 2 thành phần là mảng 1 chiều và số phần tử của nó.

- Cài đặt:**

```
const int MAX = 50; // Khai báo số phần tử tối đa của DAYSO
struct DAYSO
{
    int list[MAX];
    int n;
};
```

4. DUYỆT QUA CÁC PHẦN TỬ CỦA DÃY SỐ

Cho dãy số a có n phần tử, phần tử đầu tiên có chỉ số là 0, phần tử sau cùng có chỉ số là n-1, ta có phép duyệt các phần tử của dãy số tổng quát như sau:

```
for (int i = 0; i <= a.n-1; i++) // hoặc i < a.n
{
    // Xử lý trên phần tử a.list[i]
}
```

5. ỨNG DỤNG CÁCH DUYỆT DÃY SỐ ĐỂ NHẬP / XUẤT

- Hàm nhập:

```
void nhapDS(DAYSO &a)
{
    cout<<“\nNhap so phan tu: ”;
    cin>>a.n;
    for(int i=0; i<a.n; i++)
    {
        cout<<“a[”<<i<<“]=”;
        cin>>a.list[i];
    }
}
```

- Hàm xuất:

```
void xuatDS(DAYSO a)
{
    for(int i=0; i<a.n; i++)
        cout<< “ ” <<a.list[i];
}
```

6. THUẬT TOÁN TÍNH TỔNG, TÍCH VÀ ĐẾM

- Ý tưởng chính để tính tổng các phần tử thuộc **tập S** thỏa mãn một **điều kiện K** được thể hiện như sau :

- (1) $Kq = 0$
- (2) Duyệt mọi X thuộc S
- (3) Nếu X thỏa K thì
- (4) $Kq = Kq + X$
- (5) Cuối Duyệt

Lưu ý: Kq về phải phải có giá trị, do đó phải được gán khởi tạo ở đâu đó. Trong ví dụ này là dòng (1).

- Việc tính tích số hay đếm các phần tử của S thỏa điều kiện K được làm tương tự.
- **Để tính tích**, ta khởi tạo $Kq = 1$ (thay vì 0), và thay lệnh cộng dồn “ $Kq = Kq + X$ ” bởi lệnh nhân dồn “ $Kq = Kq * X$ ”.
- **Để đếm** ta thay “ $Kq = Kq + X$ ” bởi lệnh tăng một “ $Kq = Kq + 1$ ”
- Trường hợp K là hằng đúng (true) thì không cần kiểm tra.

Ghi nhớ: Một vòng lặp luôn phải có: khởi tạo lặp, điều kiện lặp/dừng lặp. Yếu tố làm cho điều kiện vòng lặp đúng.

III. NỘI DUNG THỰC HÀNH:

1/ Chương trình mẫu: (3 điểm)

➤ Cho đoạn mã trong file **lab2_thuvien.h** như sau:

Dòng	Mã lệnh	Ghi chú
1	#include <iostream>	
2	using namespace std;	
3		
4	const int MAX = 50; // Khai báo số phần tử tối đa	
5	//của DAYSO	
6	struct DAYSO	
7	{	
8	int list[MAX];	
9	int n;	
10	};	
11		
12	void nhapDS(DAYSO &a);	
13	void xuatDS(DAYSO a);	
14	int tongDS(DAYSO a);	

Bài 2: KIỂU DÃY SỐ - THUẬT TOÁN TÍNH TỔNG, TÍCH, ĐỀM

➤ Cho đoạn mã trong file **lab2_caidat.cpp** như sau:

Dòng	Mã lệnh	Ghi chú
1	#include "lab2_thuvien.h"	
2		
3	void nhapDS(DAYS0 &a)	
4	{	
5	cout << "\nNhap so phan tu : ";	
6	cin >> a.n;	
7	for (int i = 0; i<a.n; i++)	
8	{	
9	cout << "a[" << i << "] = ";	
10	cin >> a.list[i];	
11	}	
12		
13	}	
14		
15	void xuatDS(DAYS0 a)	
16	{	
17	for (int i = 0; i<a.n; i++)	
18	cout << a.list[i] << " ";	
19	}	
20		
21	int tongDS(DAYS0 a)	
22	{	
23	int t = 0;	
24	for (int i = 0; i < a.n; i++)	
25	t += a.list[i];	
26	return t;	
27	}	

➤ Cho đoạn mã trong file **lab2_chuongtrinh.cpp** như sau:

Dòng	Mã lệnh	Ghi chú
1	#include "lab2_thuvien.h"	
2		
3	void main()	
4	{	
5	DAYS0 d;	
6	nhapDS(d);	
7	xuatDS(d);	
8	cout << "\nTong cac phan tu cua day so = " <<	
9	tongDS(d);	
10	system("pause");	
11	}	

Yêu cầu sinh viên:

1. SV hãy nhập chương trình mẫu và thực thi chương trình.
2. Hãy sửa lại chương trình chính để đảm bảo nhập $n > 0$ (dùng cấu trúc do ... while)
3. Dựa vào thuật toán tính tổng, tích, đếm và hàm **tongDS**, SV hãy xác định: tập S là gì? X là gì? Điều kiện K là gì?
4. Từ hàm **tongDS**, SV hãy viết hàm **tichDS** để tính tích các phần tử trong dãy số.
5. Từ hàm **tongDS**, SV hãy viết hàm **demPTDuong** để đếm các phần tử dương trong dãy số.

2/ Chỉnh sửa số liệu theo yêu cầu: (2 điểm)

1. Sửa lại hàm **tongDS** để tính tổng các phần tử chẵn trong dãy số.
2. Hãy viết hàm **tichPTChiSoChan** để tính tích các phần tử có chỉ số chẵn trong dãy số.
3. SV hãy viết hàm **demPTChiaHet3Va5** để đếm các phần tử chia hết cho 3 và 5 trong dãy số.

3/ Bài tập tổng hợp: (3 điểm)

1. Viết hàm tính tổng các phần tử là số nguyên tố trong dãy số.
2. Viết hàm xuất ra màn hình số lượng các phần tử âm, số lượng các phần tử dương, số lượng các phần tử bằng 0 trong dãy số.

4/ Bài tập làm thêm: (2 điểm)

1. Viết hàm đếm số lượng số hoàn thiện trong dãy số.
2. Viết hàm kiểm tra xem dãy số có tăng dần không?
Ví dụ :
Dãy số : -6 1 4 9 12 31
Kết quả : Đây là dãy số tăng dần
3. Viết hàm cho phép thay thế phần tử x bằng phần tử y trong dãy số.
4. Viết hàm cho phép đếm số phần tử (không tính trùng nhau) trong dãy số.

I. MỤC TIÊU:

Sau khi thực hành xong bài này, sinh viên hiểu và vận dụng :

- Thuật toán tìm giá trị lớn nhất và nhỏ nhất.
- Các phép toán trên danh sách : chèn, loại bỏ, tìm kiếm, sắp xếp, trộn, tách, ghép,...

II. TÓM TẮT LÝ THUYẾT:**1. THUẬT TOÁN TÌM GIÁ TRỊ LỚN NHẤT VÀ NHỎ NHẤT:**

Bài toán đặt ra là tìm phần tử nhỏ (hay lớn nhất) trong số các phần tử của tập S thỏa mãn một điều kiện K nào đó được mô tả như sau:

Thuật toán A: nếu có thể chọn x0

```
Chọn x0 thuộc S sao cho thỏa điều kiện K và gán Min = x0
Duyệt x thuộc S (không chứa x0)
    Nếu x thỏa K và Min > x thì
        Min = X
Cuối duyệt
```

Thuật toán B: nếu “khó” chọn x0

```
Flag = false
Duyệt x thuộc S
    Nếu x thỏa K
        Nếu not Flag thì
            Min = x; Flag = true
        Ngược lại nếu Min > x thì
            Min = x;
    Cuối nếu
Cuối Duyệt
```

Nếu điều kiện K là hằng đúng (true) thì ta không cần kiểm tra, hơn nữa chỉ cần khởi gán biến Min bằng bất kỳ phần tử nào của S (thường là phần tử đầu tiên).

Trong nhiều trường hợp, ta không thể chọn được x0 bởi vì ràng buộc của điều kiện K.

Chẳng hạn tìm số dương nhỏ nhất trong mảng $a[0], a[1] \dots a[n-1]$, khởi đầu ta không thể gán Min bằng $a[0]$ hay số khác (bởi vì chưa biết số nào dương). Thuật toán B thường được áp dụng cho những trường hợp như vậy: chỉ khi nào cờ Flag được chuyển thành true thì Min mới bắt đầu xác định (lúc đó x là x_0 ở thuật toán A, nếu kết thúc thuật toán B mà Flag vẫn là false thì S không có phần tử nào thỏa điều kiện K).

2. CÁC PHÉP TOÁN TRÊN DANH SÁCH

▪ Chèn một phần tử vào danh sách

- Hàm Chèn :
 - Input :
 - a : dãy số cần chèn
 - $vitri$: vị trí cần chèn
 - x : số cần chèn
 - Output : hàm trả về 1 nếu chèn thành công, ngược lại trả về 0.
- Thuật giải :

Duyệt qua dãy số a từ phần tử cuối đến phần $[vitri]$
Gán $a.list[i] = a.list[i-1]$
Gán $a.list[i] = x$
Số phần tử của dãy tăng 1

- Mã nguồn :

```
int Chen(DAYSO &a, int vitri, int x)
{
    if (a.n == MAX)
        return 0;
    for(int i=a.n; i>vitri; i--)
        a.list[i] = a.list[i-1];
    a.list[vitri] = x; //Chèn x vào vị trí
    a.n++; //Tăng số phần tử lên 1
    return 1;
}
```

▪ Xóa phần tử

- Hàm Xóa :
 - Input :
 - a : dãy số cần xóa
 - $vitri$: vị trí cần xóa
 - Output : hàm trả về 1 nếu xóa thành công, ngược lại trả về 0.

- Thuật giải :

Nếu $\text{vitri} < 0$ hoặc $\text{vitri} \geq n$ thì trả về 0
Duyệt qua dãy số a từ phần tử $[\text{vitri}]$ đến phần tử kế cuối
Gán $a.\text{list}[i] = a.\text{list}[i+1]$
Số phần tử của dãy giảm 1

- Mã nguồn :

```
int Xoa(DAYSO &a, int vitri)
{
    if (vitri < 0 || vitri >= n) return 0;
    for(int i=vitri; i<a.n-1; i++)
        a.list[i] = a.list[i+1];
    a.n--;
    return 1;
}
```

▪ Sắp xếp

Dùng các giải thuật sắp xếp để sắp tăng dần hoặc giảm dần dãy số.

```
void BubbleSort(DAYSO &a)
{
    int i, j, x;
    for (i = 0; i < a.n - 1; i++)
        for (j = a.n - 1; j > i; j--)
            if (a.list[j] > a.list[j - 1])
            {
                x = a.list[j];
                a.list[j] = a.list[j - 1];
                a.list[j - 1] = x;
            }
}
```

▪ Tìm kiếm

- Hàm TimKiem :
 - o Input :
 - a : dãy số cần tìm
 - x : giá trị cần tìm
 - o Output : hàm trả về 1 nếu tìm thấy, ngược lại trả về 0.
- Thuật giải : Gán $\text{vitri} = -1$

Duyệt qua dãy số a từ phần tử đầu đến phần tử cuối

Nếu tìm thấy a.list[i] == x thì trả về vị trí tìm thấy và thoát vòng lặp

Trả về vitri

- Mã nguồn :

```
int TimKiem(DAYSO a, int x)
{
    int vitri = -1;
    for(int i=0; i<a.n; i++)
        if (a.list[i]==x)
        {
            vitri = i;
            break;
        }
    return vitri;
}
```

III. NỘI DUNG THỰC HÀNH:

1/ Chương trình mẫu: (3 điểm)

➤ Cho đoạn mã trong file **thuvien_lab3.h** như sau:

Dòng	Mã lệnh	Ghi chú
1	#include <iostream>	
2		
3	using namespace std;	
4		
5	const int MAX = 50;	
6	struct DAYSO	
7	{	
8	int list[MAX];	
9	int n;	
10	};	
11		
12	void nhapDS(DAYSO &a);	
13	void xuatDS(DAYSO a);	
14	int timMin(DAYSO a);	
15	int hamA(DAYSO a);	
16	void BubbleSort(DAYSO &a);	
17	DAYSO tronDS(DAYSO a, DAYSO b);	

➤ Cho đoạn mã trong file **caidat_lab3.cpp** như sau:

Dòng	Mã lệnh	Ghi chú
1	<code>#include "thuvien_lab3.h"</code>	
2		
3	<code>void nhapDS(DAYSO &a)</code>	
4	<code>{</code>	
5	<code> cout << "\nNhap so phan tu : ";</code>	
6	<code> cin >> a.n;</code>	
7	<code> for (int i = 0; i<a.n; i++)</code>	
8	<code> {</code>	
9	<code> cout << "a[" << i << "] = ";</code>	
10	<code> cin >> a.list[i];</code>	
11	<code> }</code>	
12		
13	<code>}</code>	
14		
15	<code>void xuatDS(DAYSO a)</code>	
16	<code>{</code>	
17	<code> for (int i = 0; i<a.n; i++)</code>	
18	<code> cout << a.list[i] << " ";</code>	
19	<code>}</code>	
20		
21	<code>int timMin(DAYSO a)</code>	
22	<code>{</code>	
23	<code> int min = a.list[0];</code>	
24	<code> for (int i = 0; i<a.n; i++)</code>	
25	<code> if (a.list[i]<min)</code>	
26	<code> min = a.list[i];</code>	
27	<code> return min;</code>	
28	<code>}</code>	
29		
30	<code>int hamA(DAYSO a)</code>	
31	<code>{</code>	
32	<code> int flag = 0;</code>	
33	<code> int kq;</code>	
34	<code> for (int i = 0; i<a.n; i++)</code>	
35	<code> if (a.list[i]>0)</code>	
36	<code> if (flag == 0) {</code>	
37	<code> kq = a.list[i];</code>	
38	<code> flag = 1;</code>	
39	<code> }</code>	
40	<code> else if (a.list[i]>kq)</code>	
41	<code> kq = a.list[i];</code>	
42	<code> return kq;</code>	
43	<code>}</code>	

```

44 void BubbleSort(DAYSO &a)
45 {
46     int i, j, x;
47     for (i = 0; i < a.n - 1; i++)
48         for (j = a.n - 1; j > i; j--)
49             if (a.list[j] < a.list[j - 1])
50                 {
51                     x = a.list[j];
52                     a.list[j] = a.list[j - 1];
53                     a.list[j - 1] = x;
54                 }
55 }
56 DAYSO tronDS(DAYSO a, DAYSO b)
57 {
58     DAYSO c;
59     int i = 0, j = 0, k = 0;
60     c.n = a.n + b.n;
61     while (i < a.n && j < b.n)
62     {
63         if (a.list[i] < b.list[j])
64         {
65             c.list[k] = a.list[i];
66             i++;
67         }
68         else
69         {
70             c.list[k] = b.list[j];
71             j++;
72         }
73         k++;
74     }
75     while (i < a.n)
76     {
77         c.list[k] = a.list[i];
78         i++;
79         k++;
80     }
81     while (j < b.n)
82     {
83         c.list[k] = b.list[j];
84         j++;
85         k++;
86     }
87     return c;
88 }
89

```

➤ Cho đoạn mã trong file **chuongtrinh_lab3.cpp** như sau:

Dòng	Mã lệnh	Ghi chú
1	<code>#include "thuvien_lab3.h"</code>	
2		
3	<code>void main()</code>	
4	<code>{</code>	
5	<code> DAYSO d;</code>	
6	<code> nhapDS(d);</code>	
7	<code> xuatDS(d);</code>	
8	<code> cout << "\nMin la: " << timMin(d);</code>	
9	<code> int kq = hamA(d);</code>	
10	<code> cout << "\nHamA la: " << kq;</code>	
11		
12	<code> DAYSO a1, a2, a3;</code>	
13	<code> cout << "\nNhap day so thu 1:";</code>	
14	<code> nhapDS(a1);</code>	
15	<code> xuatDS(a1);</code>	
16	<code> cout << "\nNhap day so thu 2:";</code>	
17	<code> nhapDS(a2);</code>	
18	<code> xuatDS(a2);</code>	
19	<code> cout << "\nDay so thu 1 sap tang dan:";</code>	
20	<code> BubbleSort(a1);</code>	
21	<code> xuatDS(a1);</code>	
22	<code> cout << "\nDay so thu 2 sap tang dan:";</code>	
23	<code> BubbleSort(a2);</code>	
24	<code> xuatDS(a2);</code>	
25	<code> a3 = tronDS(a1, a2);</code>	
26	<code> cout << "\nDay so da duoc tron tang dan:";</code>	
27	<code> xuatDS(a3);</code>	
28	<code> system("pause");</code>	
29	<code>}</code>	

Yêu cầu sinh viên:

1. SV hãy nhập chương trình mẫu và thực thi chương trình.
2. Cho biết ý nghĩa của **hamA** ?
3. **hamA** đã sử dụng giải thuật nào? Cho biết các đại lượng: tập S, x0, điều kiện K.
4. Nếu muốn sửa **hamA** lại tìm số chẵn nhỏ nhất thì sửa như thế nào?
5. Nếu muốn hàm **BubbleSort** chuyển thành sắp xếp giảm dần thì đổi như thế nào?
6. Cho biết ý nghĩa và giải thích các tham số của hàm **tronDS** ?
7. Giải thích vòng lặp while từ dòng lệnh 63 đến 76?
8. Giải thích vòng lặp while từ dòng lệnh 77 đến 82?
9. Giải thích vòng lặp while từ dòng lệnh 83 đến 88?

2/ **Chỉnh sửa số liệu theo yêu cầu: (2 điểm)**

1. Hãy sửa lại **hamA** để tìm phần tử chia hết cho 3 lớn nhất trong dãy số.
2. Viết hàm **tách** dãy số thành 2 dãy số tại vị trí index.

Ví dụ :

Dãy số : 3 -6 8 1 9 2 1

Index = 3 Kết quả :

Dãy số 1 : 3 -6 8

Dãy số 2 : 1 9 2 1

3/ **Bài tập tổng hợp: (3 điểm)**

1. Viết hàm **ghép** 2 dãy số thành 1 dãy số. Ví dụ :

Dãy số 1 : 3 -6 8

Dãy số 2 : 1 9 2 1

Kết quả : 3 -6 8 1 9 2 1

2. Viết hàm **đảo ngược** dãy số. Ví dụ :

Dãy số : 3 -6 8 1 9 2 1

Kết quả :

Dãy số : 1 2 9 1 8 -6 3

3. Viết hàm **xóa tất cả** phần tử có giá trị bằng x trong dãy số.

Ví dụ :

Dãy số : 3 -6 8 1 9 2 1

Nhập x = 1 Kết quả :

Dãy số : 3 -6 8 9 2

4/ **Bài tập làm thêm: (2 điểm)**

1. Viết hàm xóa tất cả phần tử lớn nhất trong dãy số.
2. Viết hàm xóa tất cả các phần tử trùng nhau trong dãy số
3. Viết hàm sắp xếp lại dãy số sao cho các phần tử chẵn tăng dần ở đầu dãy số, các phần tử lẻ giảm dần ở cuối dãy số.
4. Viết hàm ghép dãy số 2 vào sau dãy số 1.
5. Viết hàm tách dãy số ban đầu thành 2 dãy số: dãy số 1 gồm các phần tử chẵn, dãy số 2 gồm các phần tử lẻ.
6. Viết chương trình tính các phép toán + , - , * , / cho số nguyên dài.

Gợi ý : mỗi số trong số nguyên dài ta đưa vào 1 phần tử trong mảng. VD :

4	2	3	0	2	0	7	9	4	3	0
---	---	---	---	---	---	---	---	---	---	---

=> số nguyên có chiều dài 11 số đưa vào 11 phần tử của mảng.

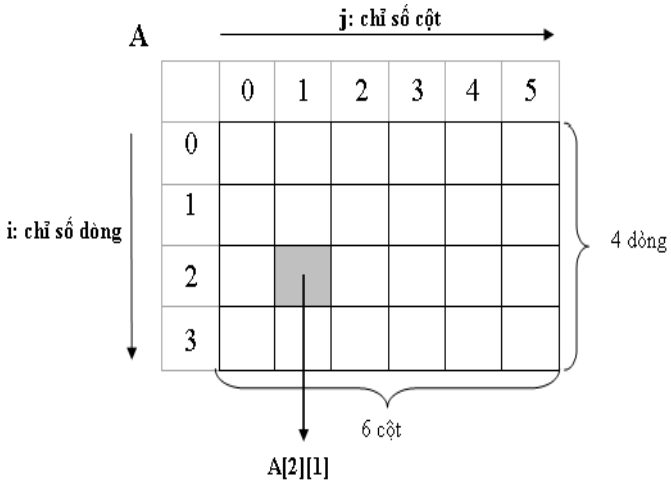
=> các phép toán sẽ tính dựa trên 2 dãy số nguyên.

I. MỤC TIÊU:

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

- Khai báo mảng 2 chiều
- Đọc file, ghi file, đóng file
- Phép duyệt ma trận
- Duyệt dòng, cột của ma trận

II. TÓM TẮT LÝ THUYẾT:

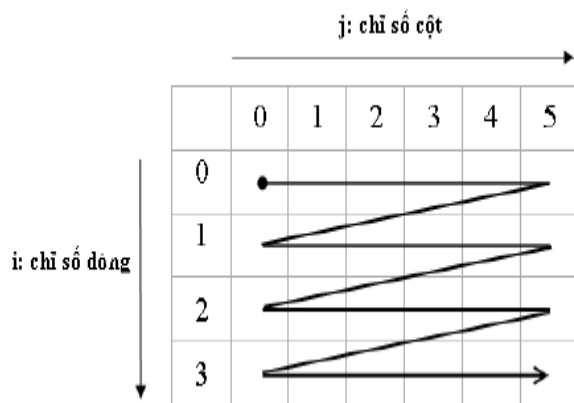
Nội dung	Ví dụ
1. Ma trận vuông	
Khái niệm: <ul style="list-style-type: none"> - Ma trận vuông là trường hợp đặc biệt của ma trận, có số dòng = số cột 	
Khai báo: <kiểu DL><Tên Mảng>[số dòng][số cột]	<pre>int A[10][20]</pre> <ul style="list-style-type: none"> - Khai báo mảng A có 10 dòng, 20 cột. Có tối đa 10x20 phần tử. <pre>float B[5][5]</pre> <ul style="list-style-type: none"> - Khai báo mảng B có 5 dòng, 5 cột. Có tối đa 5x5 phần tử.
2. Xây dựng kiểu ma trận Ý tưởng: Ta thấy đối với mảng hai chiều, muốn duyệt mảng phải biết được số dòng và số cột. Vì vậy, để cho đơn giản, ta tạo ra kiểu dữ liệu MATRAN có 3 thành phần là mảng 2 chiều, số dòng và số cột của nó.	

Cài đặt:

```
const int MAX = 50; //khai báo tối đa số phần tử
```

```
struct MATRAN{
    int list[MAX][MAX];
    int n; //số dòng
    int m; //số cột
};
```

Duyệt các phần tử của ma trận:



Cài đặt:

```
for(int i=0;i<a.n;i++){
    for(int j=0;j<a.m;j++){
        //Duyệt phần tử a.list[i][j]
    }
}
```

Duyệt các phần tử theo dòng:

Gọi i là dòng muốn duyệt các phần tử, ta có cách duyệt sau:

```
for (int i=0;i<a.n;i++)
    //Duyệt phần tử a.list[i][j]
```

Duyệt các phần tử theo cột:

Gọi j là cột muốn duyệt các phần tử, ta có cách duyệt sau:

```
for (int j=0;j<a.m;j++)
    //Duyệt phần tử a.list[i][j]
```

3. Đọc file, ghi file văn bản

- **Mở file:**

- Khai báo con trỏ file:

```
FILE* fp; //con trỏ fp kiểu FILE
```

- Mở file

```
fp= fopen(path, mode);
```

- path: đường dẫn đến file muốn mở
- mode: chế độ mở gồm các chế độ sau:

	<ul style="list-style-type: none">▪ “r”: mở chế độ đọc, file phải tồn tại▪ “w”: tạo file mới để ghi. Nếu file có sẵn thì bị ghi đè.▪ “a”: mở để ghi tiếp nội dung, nếu file không có sẵn thì tạo file mới. <p>Hàm trả về con trỏ file nếu mở được file, trả về NULL nếu không mở được file.</p>
<p>- Cách lưu trữ ma trận trong file</p>	<ul style="list-style-type: none">- Dòng 1: Lưu trữ số dòng và cột của ma trận- Các dòng kế tiếp: lưu trữ các phần tử của ma trận- Ví dụ: Lưu trữ ma trận 3 dòng, 4 cột trong file <div><div><div>34</div><div>1234</div><div>5678</div><div>9101112</div></div><div><div>← số dòng và cột của ma trận</div><div>} Các phần tử của ma trận</div></div></div>
<p>- Đọc file</p>	<p>fscanf(fp, “chuỗi đặc tả”, [&biến 1, &biến 2...];</p> <ul style="list-style-type: none">- fp: là con trỏ file- “chuỗi đặc tả”: %d: số nguyên int, %f: số thực, %ld: là số nguyên long, %c: ký tự- Biến 1, biến 2: là các biến muốn đọc giá trị từ file <p>➤ Đọc ma trận từ file</p> <ul style="list-style-type: none">- Hàm load- Input: filename: tên tập tin cần đọc- Output:<ul style="list-style-type: none">• a: ma trận đọc được từ filename• Hàm trả về 1 nếu đọc dữ liệu thành công, ngược lại trả về 0. <p>➤ Mã nguồn:</p> <pre>int loadMaTran(MATRAN &a, char *filename) { FILE *fp; fp = fopen(filename, "r"); if (fp != NULL){ fscanf(fp, "%d", &a.n); //đọc số dòng fscanf(fp, "%d", &a.m); //đọc số cột //đọc các phần tử của ma trận for (int i = 0; i < a.n; i++)</pre>

	<pre> for (int j = 0; j < a.m; j++) fscanf(fp, "%d", &a.list[i][j]); fclose(fp); //đóng tập tin return 1; } return 0; } </pre>
<p>- Ghi file</p>	<p>fprint(fp,"chuỗi đặc tả",[biểu thức 1,biểu thức 2,...]);</p> <ul style="list-style-type: none"> - fp: là con trỏ file - “Chuỗi đặc tả”: %d: số nguyên int, %f: số thực, %ld: số nguyên long, %c: ký tự - Biểu thức 1, biểu thức 2: là các giá trị của biểu thức muốn ghi ra file <p>➤ Hàm Save:</p> <ul style="list-style-type: none"> • Input: a ma trận cần lưu vào tập tin • Output: filename: tên tập tin cần lưu ma trận <p>➤ Mã nguồn:</p> <pre> void saveMaTran(MATRAN &a, char *filename) { FILE *fp; fp = fopen(filename, "w"); if (fp != NULL) { fprintf(fp, "%d", a.n); //Ghi số dòng fprintf(fp, "%d", a.m); //Ghi số cột //Ghi các phần tử của ma trận for (int i = 0; i < a.n; i++){ fprintf(fp, "\n"); //Xuống hàng for (int j = 0; j < a.m; j++){ fprintf(fp, "%d", a.list[i][j]); fprintf(fp, "\t"); } } fclose(fp); } } </pre>

III. NỘI DUNG THỰC HÀNH

1/ Chương trình mẫu

Câu 1: Khai báo cấu trúc MATRAN. Viết hàm **Nhap**, **Xuat**, hàm **tinhTong** tính tổng các phần tử của ma trận. Chương trình nhập và ma trận cấp mxn, xuất ma trận và tính tổng của ma trận đó ra màn hình.

- Tập tin thư viện như sau **thuvien.h**:

Mã lệnh
<pre>#include<iostream> using namespace std; const int MAX = 50; struct MATRAN{ int list[MAX][MAX]; int n; int m; }; void nhapMaTran(MATRAN &a); void xuatMaTran(MATRAN &a); int tinhTong(MATRAN a);</pre>

- Tập tin cài đặt như sau **caidat.cpp**:

Mã lệnh
<pre>#include"thuvien.h" void nhapMaTran(MATRAN &a){ cout << "\n Nhap so dong:"; cin >> a.n; cout << "\n Nhap so cot:"; cin >> a.m; for (int i = 0; i < a.n; i++){ for (int j = 0; j < a.m; j++){ cout << "a[" << i << "]" << "[" << j << "]="; cin >> a.list[i][j]; } } void xuatMaTran(MATRAN &a){ cout << "Mang vua nhap:"; for (int i = 0; i < a.n; i++){ cout << endl; for (int j = 0; j < a.m; j++){ cout << a.list[i][j] << " "; } } } int tinhTong(MATRAN a){ int s = 0;</pre>

```
    for (int i = 0; i < a.n; i++)
        for (int j = 0; j < a.m; j++)
            s += a.list[i][j];
    return s;
}
```

- File chương trình **chuongtrinh.cpp**:

Mã lệnh
<pre>#include"thuvien.h" void main(){ MATRAN a; nhapMaTran(a); xuatMaTran(a); int tong = tinhTong(a); cout << "\n Tong cac phan tu:" << tong; system("pause"); }</pre>

Câu 2: Đọc vào một ma trận 2 chiều từ tập tin, xuất tập tin ra, đếm các phần tử dương có trong ma trận vừa đọc, viết hàm nhân ma trận vừa đọc với 2 và lưu lại ma trận đó.

- Tập tin thư viện như sau **thuvien.h**:

Mã lệnh
<pre>#define _CRT_SECURE_NO_WARNINGS #include<iostream> using namespace std; const int MAX = 50; struct MATRAN{ int list[MAX][MAX]; int n; int m; }; int loadMaTran(MATRAN &a, char *filename); void inMaTran(MATRAN a); int demDuong(MATRAN a); void nhan_MaTran(MATRAN &a); void save_MaTran(MATRAN &a, char *filename);</pre>

- Tập tin cài đặt như sau **caidat.cpp**:

Mã lệnh

```
int loadMaTran(MATRAN &a, char *filename){
    FILE *fp;
    fp = fopen(filename, "r");
    if (fp != NULL)
    {
        fscanf(fp, "%d", &a.n);
        fscanf(fp, "%d", &a.m);
        for (int i = 0; i < a.n; i++)
            for (int j = 0; j < a.m; j++)
                fscanf(fp, "%d", &a.list[i][j]);
        fclose(fp);
        return 1;
    }
    return 0;
}

void inMaTran(MATRAN a){
    for (int i = 0; i < a.n; i++)
    {
        cout << "\n";
        for (int j = 0; j < a.m; j++)
            cout << a.list[i][j] << "\t";
    }
}

int demDuong(MATRAN a){
    int dem = 0;
    for (int i = 0; i < a.n; i++)
        for (int j = 0; j < a.m; j++)
            if (a.list[i][j] > 0)
                dem++;
    return dem;
}

void nhan_MaTran(MATRAN &a){
    int x;
    cout << "\n Nhap so can nhan:";
    cin >> x;
    for (int i = 0; i < a.n; i++)
        for (int j = 0; j < a.m; j++)
            a.list[i][j] *= x;
}

void save_MaTran(MATRAN &a, char *filename){
```

```
FILE *fp;
fp = fopen(filename, "w");
if (fp != NULL)
{
    fprintf(fp, "%d", a.n);
    fprintf(fp, "\t");
    fprintf(fp, "%d", a.m);
    for (int i = 0; i < a.n; i++)
    {
        fprintf(fp, "\n");
        for (int j = 0; j < a.m; j++)
        {
            fprintf(fp, "%d", a.list[i][j]);
            fprintf(fp, "\t");
        }
    }
}
```

- File chương trình **chuongtrinh.cpp**:

Mã lệnh

```
#include "thuvien.h"
void main(){
    if (loadMaTran(a, "E:\\matran.txt") == 1)
    {
        inMaTran(a);
        cout << "\n Cac so duong:" << demDuong(a);
        nhan_MaTran(a);
        inMaTran(a);
        save_MaTran(a, "E:\\mt.txt");
    }
    else
        cout << "khong doc duoc file";
        system("pause");
}
```

2/ Chỉnh sửa số liệu theo yêu cầu

Câu 1: Sửa hàm **tínhTong** ở trên, để tính tổng các phần tử Chẵn có trong ma trận.

Câu 2: Sửa hàm **demDuong** ở trên để đếm các số lẻ có trong ma trận

3/ Vận dụng

Câu 1: Hãy viết hàm tìm phần tử nhỏ nhất có trong ma trận

Câu 2: Cho ma trận cấp $m \times n$, tính tổng các dòng có trong ma trận

Hướng dẫn:

3	2	5
1	7	-4
9	-8	11

- Dòng 1 tổng = 10
- Dòng 2 tổng = 4
- Dòng 3 tổng = 12

Câu 3: Tìm dòng có tổng lớn nhất. Ví dụ ở trên dòng có tổng lớn nhất là dòng 3.

4/ Bài tập tổng hợp về nhà

Câu 1: Viết hàm tính tổng các phần tử lớn nhất trên mỗi dòng/

Câu 2: Tính tổng các cột có trong ma trận.

I. MỤC TIÊU:

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

- Thực hiện các thao tác trên ma trận vuông (duyet đường chéo chính, đường chéo phụ, các phép tính có trên ma trận vuông, ...)
- Thuật toán cho lượng từ tồn tại và với mọi

II. TÓM TẮT LÝ THUYẾT:

Nội dung	Ví dụ									
1. Ma Trận vuông										
Khái niệm: <ul style="list-style-type: none">- Ma trận vuông là trường hợp đặc biệt của ma trận có số dòng = số cột. Ta có thể xây dựng kiểu MATRANVUONG gồm 2 thành phần gồm: mảng 2 chiều, cấp của ma trận vuông.	<ul style="list-style-type: none">- Ma trận vuông cấp 3, có tối đa 3x3 phần tử <div><table><tr><td>5</td><td>4</td><td>8</td></tr><tr><td>-9</td><td>12</td><td>1</td></tr><tr><td>9</td><td>-12</td><td>15</td></tr></table></div>	5	4	8	-9	12	1	9	-12	15
5	4	8								
-9	12	1								
9	-12	15								
Khai báo: <kiểu DL><Tên Mảng>[số dòng][số cột]	int A[10][10] <ul style="list-style-type: none">- Khai báo ma trận vuông A có 10 dòng, 10 cột. Có tối đa 10x10 phần tử. float B[5][5] <ul style="list-style-type: none">- Khai báo ma trận vuông B có 5 dòng, 5 cột. Gọi là ma trận vuông cấp 5. Có tối đa 5x5 phần tử.									
2. Xây dựng kiểu ma trận										
Ý tưởng: Ta thấy đối với ma trận vuông, muốn duyệt ma trận phải biết được cấp của ma trận. Vì vậy, để đơn giản, ta tạo ra kiểu dữ liệu MATRANVUONG có 2 thành phần là mảng 2 chiều, cấp của ma trận.										

Cài đặt:

```
const int MAX = 50; //khai báo tối đa số phần tử
```

```
struct MATRANVUONG{
```

```
    int list[MAX][MAX];
```

```
    int n; //cấp của ma trận
```

```
};
```

- **Duyệt các phần tử trên đường chéo chính:**

Các phần tử trên đường chéo chính có chỉ số dòng bằng chỉ số cột: $a[i][i]$

Cài đặt:

```
for(int i=0;i<a.n;i++)  
    //Duyệt phần tử a.list[i][i]
```

- **Duyệt các phần tử trên đường chéo phụ**

Các phần tử trên đường chéo phụ có chỉ số dòng là i và chỉ số cột là $n - i - 1$: $a[i][n-i-1]$

```
for (int i=0;i<a.n;i++)  
    //Duyệt phần tử a.list[i][a.n-i-1]
```

- **Lượng tử tồn tại và với mọi**

Một trong những thủ thuật đóng vai trò quan trọng trong lập trình là làm sao cài đặt việc kiểm tra các khẳng định có chứa lượng tử tồn tại hay với mọi. Chẳng hạn khẳng định một số nguyên $n > 1$ không phải là số nguyên tố được viết bởi mệnh đề: “Tồn tại số tự nhiên m sao cho $1 < m < n$ và m là ước của n ”. Sau đây là thuật toán tổng quát cho loại mệnh đề này (Trong đó $P(x)$ là một khẳng định tùy thuộc và một biến x nào đó):

- **Cài đặt kiểm tra tồn tại:**
 $(\exists x \in S) (P(x))$

```
KT := false
```

```
Duyệt  $x \in S$ 
```

```
    Nếu  $P(x)$ thỏa thì
```

```
        KT := true
```

```
        Dừng quá trình duyệt S
```

```
    Cuối nếu
```

```
    Cuối duyệt
```


<ul style="list-style-type: none">- Cài đặt kiểm tra với mọi $\forall (\forall x \in S) (P(x))$	<pre>KT := true Duyệt x ∈ S Nếu not P(x) thì KT := false Dừng quá trình duyệt S Cuối nếu Cuối duyệt</pre>
<ul style="list-style-type: none">- Cả hai thuật toán trên: giá trị của biến KT khi thuật toán dừng, chính là chân trị của mệnh đề cần kiểm tra, giá trị này thường được cài đặt như giá trị trả về của hàm khi sử dụng một ngôn ngữ lập trình cụ thể.	

III. NỘI DUNG THỰC HÀNH

1/ Chương trình mẫu

Câu 1: Đoạn mã sau, dùng để nhập, xuất ra một ma trận vuông. Có 2 hàm: **xuatCheoChinh** dùng để xuất các phần tử có trên đường chéo chính và hàm **tingTongCheoPhu** dùng để tính tổng các phần tử có trên đường chéo phụ.

- Tập tin `thuvien.h` như sau:

Mã lệnh
<pre>#include<iostream> using namespace std; const int MAX = 50; struct MTVUONG{ int list[MAX][MAX]; int n; }; void nhapMTVuong(MTVUONG &a); void xuatMTVuong(MTVUONG a); void xuatCheoChinh(MTVUONG a); int tingTongCheoPhu(MTVUONG a);</pre>

- Tập tin cài đặt như sau:

Mã lệnh
<pre>#include"thuvien.h" void nhapMTVuong(MTVUONG &a){ cout << "\n Nhập cap cua ma tran:"; cin >> a.n; for (int i = 0; i < a.n; i++) for (int j = 0; j < a.n; j++){</pre>

```
        cout << "a[" << i << "], " << j << "]=";
        cin >> a.list[i][j];
    }
}

void xuấtMTVuong(MTVUONG a){
    cout << "\n Ma tran vua nhap:";
    for (int i = 0; i < a.n; i++){
        cout << endl;
        for (int j = 0; j < a.n; j++)
            cout << a.list[i][j] << "\t";
    }
}

void xuấtCheoChinh(MTVUONG a){
    for (int i = 0; i < a.n; i++)
        cout << a.list[i][i] << "\t";
}

int tínhTongCheoPhu(MTVUONG a){
    int s = 0;
    for (int i = 0; i < a.n; i++)
        s += a.list[i][a.n - i - 1];
    return s;
}
```

- Tập tin chương trình như sau:

Mã lệnh
<pre>#include"thuvien.h" void main(){ MTVUONG a; nhapMTVuong(a); xuấtMTVuong(a); xuấtCheoChinh(a); cout << "\n Tong cac phan tu tren cheo phu:"<< tínhTongCheoPhu(a); system("pause"); }</pre>

Câu 2: Đoạn mã sau có 2 hàm, hàm **bool kiểmTraLe** dùng để kiểm tra mọi phần tử có trong ma trận toàn là số lẻ hay không, nếu có trả về true, nếu không trả về false. Hàm **bool kiểmTra_Boi3** dùng để kiểm tra có tồn tại phần tử là bội của 3 trên đường chéo phụ hay không, nếu tồn tại thì trả về true, ngược lại trả về false.

- Tập tin thư viện bổ sung vào câu 1 như sau:

Mã lệnh

```
bool kiemTraLe(MTVUONG a);  
bool kiemTra_Boi3(MTVUONG a);
```

- Tập tin cài đặt bổ sung thêm vào câu 1:

Mã lệnh

```
bool kiemTraLe(MTVUONG a){  
    bool kt = true;  
    for (int i = 0; i < a.n; i++)  
        for (int j = 0; j < a.n; j++)  
            if (a.list[i][j] % 2 == 0)  
                kt = false;  
    return kt;  
}  
bool kiemTra_Boi3(MTVUONG a){  
    bool kt = false;  
    for (int i = 0; i < a.n; i++)  
        if (a.list[i][a.n - i - 1] % 3 == 0)  
            kt = true;  
    return kt;  
}
```

- Tập tin chương trình bổ sung vào như sau:

Mã lệnh

```
#include "thuvien.h"  
void main(){  
    MTVUONG a;  
    nhapMTVuong(a);  
    xuatMTVuong(a);  
    xuatCheoChinh(a);  
    cout << "\n Tong cac phan tu tren cheo phu:" <<  
    tinhTongCheoPhu(a);  
    if (kiemTraLe(a) == true)  
        cout << "\n Cac phan tu trong ma tran la so  
Le";  
    else  
        cout << "\n Cac phan tu trong ma tran ko hoan  
toan la so le";  
  
    if (kiemTra_Boi3(a) == true)
```

```
        cout << "\n ton tai phan tu tren duong cheo  
phu la Boi cua 3";  
    else  
        cout << "\n Khong co phan tu Boi 3 tren duong  
cheo Phu";  
    system("pause");  
}
```

Câu 3: Cho đoạn mã dùng để in tam giác trái dưới như sau:

Mã lệnh
<pre>void inTamGiacTraiDuoi(MTVUONG a){ for (int i = 0; i < a.n; i++){ for (int j = 0; j < a.n; j++){ if (j <= i) cout << a.list[i][j] << "\t"; cout << endl; } } }</pre>

2/ Chỉnh sửa số liệu theo yêu cầu

Câu 1: Yêu cầu viết hàm in tam giác trên trái

Ví dụ: Cho ma trận vuông cấp 3 như sau:

1	2	3
4	5	6
7	8	9



1	2	3
4	5	
7		

Câu 2: Bổ sung vào chương trình, yêu cầu viết hàm in tam giác phía trên bên phải

Ví dụ:

1	2	3
4	5	6
7	8	9



1	2	3
	5	6
		9

Câu 3: Bổ sung vào chương trình, yêu cầu viết hàm in tam giác phía dưới bên phải

Ví dụ:

1	2	3
4	5	6
7	8	9



		3
	5	6
7	8	9

Câu 4: Viết hàm kiểm tra xem ở trong tam giác phía dưới bên phải, tồn tại phần tử là số âm hay không? Nếu có trả về true, ngược lại trả về false.

Bài 5: MA TRẬN – CÁC PHÉP TOÁN TRÊN MA TRẬN

Câu 5: Viết hàm kiểm tra xem ở tam giác phải trên tất cả các phần tử là số chẵn hay không? Nếu tất cả là số chẵn thì trả về true, ngược lại trả về false.

3/ Vận dụng

Câu 1: Viết hàm nhập thêm vào ma trận thứ 2. Viết hàm tính tổng hai ma trận.

1	2	3
4	5	6
7	8	9

 +

-5	9	3
12	15	-6
17	20	14


 =

-4	11	6
16	20	0
24	28	23

Câu 2: Viết hàm in ma trận chuyển vị của các phần tử qua đường chéo chính.

Ví dụ:

1	2	3
4	5	6
7	8	9




1	4	7
2	5	8
3	6	9

Câu 3: Viết hàm chuyển đổi các phần tử qua đường chéo phụ

Ví dụ:

1	2	3
4	5	6
7	8	9



9	6	7
8	5	2
3	4	1

4/ Bài tập tổng hợp về nhà.

Câu 1: Tính tổng hai ma trận

Câu 2: Tính tích hai ma trận

Câu 3: Tính tổng các phần tử lớn nhất trên mỗi dòng có trong ma trận.

Câu 4: Kiểm tra xem ma trận nhập vào có phải là ma trận đơn vị hay không?

Gợi ý: Ma trận đơn vị là ma trận mà các phần tử trên đường chéo chính bằng 1, các phần tử khác bằng 0.

Câu 5: Kiểm tra ma trận nhập vào có phải là ma trận đối xứng không. (ma trận đối xứng khi và chỉ khi có $a_{ij} = a_{ji}$ với mọi i, j).

I. MỤC TIÊU:

Sau khi thực hành xong bài này, sinh viên có thể :

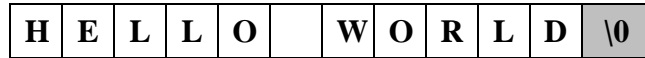
- Hiểu được con trỏ và cấp phát động.
- Biết các sử dụng các hàm xử lý chuỗi trong thư viện **string.h** (strcat, strcpy, strlen, strcmp, strchr, strstr, strlwr,strupr, ...).
- Ứng dụng cho các bài toán xử lý chuỗi.

II. TÓM TẮT LÝ THUYẾT:**1) Con trỏ và cấp phát động:**

- Con trỏ là một loại biến đặc biệt không dùng để chứa giá trị mà để chứa địa chỉ của một biến.
- **Khai báo:**
<kiểu dữ liệu> *<ten_bien>;
Vd: int *pN; //con trỏ kiểu int
char *p; // con trỏ kiểu char
- **Toán tử &:**
Dùng để lấy địa chỉ của một biến.
int n = 5;
int *pn = &n; //Khai báo biến con trỏ kiểu int giữ địa chỉ biến n
- **Cấp phát vùng nhớ:**
Dùng từ khóa **new** để xin cấp phát vùng nhớ. Sau khi sử dụng xong, phải thu hồi vùng nhớ đã xin cấp phát.
Vd: int *pn; //Khai báo biến con trỏ kiểu int, chưa có vùng nhớ.
pn = **new** int; //Xin cấp phát vùng nhớ kiểu int do biến pn trỏ đến.
- **Giải phóng vùng nhớ:**
Dùng từ khóa **delete** để thu hồi vùng nhớ được cấp phát bởi từ khóa new.
Vd: **delete** pn;

2) Chuỗi:

- Chuỗi là mảng 1 chiều gồm có các phần tử là kiểu char và được kết thúc bởi ký tự NULL ('\\0')



↑
Ký tự NULL

- **Khai báo:**

`char <ten_chuoi>[số ký tự + 1];`

- **Truy cập các phần tử trong chuỗi:**

`<ten_chuoi>[chỉ số]` (từ 0 đến chiều dài chuỗi -1)

- **Nhập / Xuất chuỗi:**

- Nhập chuỗi: dùng **`cin.getline(<ten_chuoi>, <số ký tự + 1>);`**
- Bỏ qua ký tự trên dòng nhập: **`cin.ignore(<số ký tự>);`** Lệnh này thường được sử dụng để bỏ qua ký tự Enter trên dòng nhập.

III. NỘI DUNG THỰC HÀNH:**1/ Chương trình mẫu: (4 điểm)**

Cho 3 file chương trình mẫu: **thuvien_lab6.h**, **caidat_lab6.cpp**, **chuongtrinh_lab6.cpp** có nội dung như sau:

```
thuvien_lab6.h  X
Lab6
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <iostream>
3  const int MAX = 100;
4  using namespace std;
5
6  void NhapChuoi(char *s);
7  char* DaoChuoi(char *s);
8  char* ChenChuoi(char *s, char *sub, int pos);
9  void hoanvi(char &x, char &y);
```

```
caidat_lab6.cpp X
Lab6
1  #include "thuvien_lab6.h"
2  void NhapChuoi(char *s) {
3      cin.getline(s, MAX + 1);
4  }
5  void hoanvi(char &x, char &y) {
6      char temp = x;
7      x = y;
8      y = temp;
9  }
10 char* DaoChuoi(char *s) {
11     char *t = new char[MAX + 1];
12     strcpy(t, s);
13     int i = 0;
14     int j = strlen(t) - 1;
15     while (i < j) {
16         hoanvi(t[i], t[j]);
17         i++;
18         j--;
19     }
20     return t;
21 }
22 char* ChenChuoi(char *s, char *sub, int pos) {
23     char *t;
24     if (pos < 0) pos = 0;
25     if (pos > strlen(s)) pos = strlen(s);
26     t = new char[strlen(s + pos) + 1];
27     strcpy(t, s + pos);
28     s[pos] = '\0';
29     strcat(s, sub);
30     strcat(s, t);
31     return s;
32 }
```



```
chuongtrinh_lab6.cpp X
Lab6
1  #include "thuvien_lab6.h"
2  int main() {
3      char s[MAX];
4      char *t;
5      cout << "\nNhap chuoi:";
6      NhapChuoi(s);
7      cout << "\nChuoi vua nhap: " << s << " ";
8      t = DaoChuoi(s);
9      cout << "\n\nChuoi dao nguoc: " << t << " ";
10
11     char s1[MAX], sub[MAX];
12     int pos;
13     cout << "\nNhap chuoi chinh: ";
14     NhapChuoi(s1);
15     cout << "\nNhap chuoi can chen: ";
16     NhapChuoi(sub);
17     cout << "\nNhap vi tri can chen: ";
18     cin >> pos;
19     t = ChenChuoi(s1, sub, pos);
20     cout << "\nChuoi da chen: " << t;
21     cout << endl;
22     system("pause");
23     return 0;
24 }
```

Người dùng sẽ nhập một chuỗi từ bàn phím, in ra:

- Chuỗi vừa nhập.
- Chuỗi đảo ngược của chuỗi vừa nhập.
- Thực hiện việc chèn một chuỗi sub vào chuỗi s tại vị trí pos.

Yêu cầu sinh viên:

- Biên dịch chương trình trên.
- Cho biết kết quả in ra màn hình khi người dùng nhập dữ liệu sau: “đại học STU”
- Nếu khai báo hàm NhapChuoi như sau được không? Giải thích.

```
void NhapChuoi(char s[MAX]);
```

- Mở file caidat_lab6.cpp, trả lời các câu hỏi sau:

Dòng 3: tại sao MAX+1

.....
.....
.....

Dòng 10: công dụng hàm, tham số, giá trị trả về của hàm

.....

.....

.....

Dòng 12: công dụng của lệnh này.

.....

.....

.....

Dòng 15-19: giải thích vòng lặp while

.....

.....

.....

Dòng 22: công dụng hàm, tham số, giá trị trả về của hàm

.....

.....

.....

Dòng 25: công dụng của lệnh này.

.....

.....

.....

Dòng 26: công dụng của lệnh này.

.....

.....

.....

Dòng 27: công dụng của lệnh này.

.....

.....

.....

Dòng 28: công dụng của lệnh này.

.....

.....

.....

Dòng 29: công dụng của lệnh này.

.....

.....

.....

Dòng 30: công dụng của lệnh này.

.....

.....

.....

Dòng 31: công dụng của lệnh này.

.....

.....

- e) Trong hàm DaoChuoi, hãy thay vòng lặp while thành for
- f) Trong hàm DaoChuoi (file caidat_lab6, dòng 11), ta khai báo:
char t[MAX] thay **char *t** được không?
- g) Trong hàm ChenChuoi, ta truyền tham số pos là số âm được không? Giải thích.

2/ Chỉnh sửa số liệu theo yêu cầu: (0.2 điểm)

- a) Bổ sung chương trình mẫu cho phép đổi chữ thành chữ HOA (không dùng hàmstrupr)
VD: Nhập “cong nghe thong tin” → “CONG NGHE THONG TIN”
- b) Bổ sung chương trình mẫu cho phép in ra số các ký tự chữ HOA, số các ký tự chữ thường, số các ký tự chữ số.
VD: Nhập “Cong Nghe Thong Tin 2020”
→ Kết quả:
Số ký tự chữ HOA: 4
Số ký tự chữ thường: 12
Số ký tự chữ số: 4
- c) Bổ sung chương trình mẫu cho phép đổi chữ HOA thành chữ thường, chữ thường thành chữ HOA, các ký tự khác giữ nguyên.
VD: Nhập “Cong Nghe Thong Tin 2020”
→ Kết quả: “cONG nGHE tHONG tIN 2020”

3/ Bài tập tổng hợp: (0.2 điểm)

- a) Bổ sung chương trình mẫu cho phép viết HOA đầu mỗi từ.
VD: Nhập “cong nghe thong tin”
→ Kết quả: “Cong Nghe Thong Tin”
- b) Tìm tất cả vị trí xuất hiện của chuỗi con Sub trong chuỗi mẹ S.
VD: Nhập “abcdefabcdefabc”
→ Kết quả: Vị trí của “abc” là: 0 6 12
- c) Thực hiện xóa num ký tự trong chuỗi s bắt đầu từ vị trí pos
VD: Nhập “Cong Nghe Thong Tin”
Pos = 5

Num = 5

→Kết quả: “Cong Thong Tin”

4/ Bài tập làm thêm: (0.2 điểm)

- a) Trích một chuỗi con từ chuỗi mẹ S với vị trí trích là pos và trích num ký tự.

VD: Nhập “Cong Nghe Thong Tin”

Pos = 5

Num = 4

→Kết quả: “Nghe”

- b) Bổ sung chương trình mẫu cho phép tìm kiếm và thay thế chuỗi con bằng chuỗi thay thế trong chuỗi mẹ.

VD: Nhập “abcdefabcbgth”

Chuỗi tìm kiếm: “abc”

Chuỗi thay thế: “A”

→Chuỗi kết quả: “AdefAgth”

- c) Bổ sung chương trình mẫu cho phép chuẩn hóa chuỗi. (xóa khoảng trắng đầu và cuối chuỗi, xóa những khoảng trắng dư thừa giữa các từ).

VD: Chuỗi mẹ: “ ab c de fg ”

→Chuỗi đã chuẩn hóa: “ab c de fg”

- d) Thực hiện việc tách một chuỗi họ tên thành họ lót và tên theo tên người Việt Nam. Ví dụ: họ tên = “Lê Triệu Ngọc Đức” → họ lót = “Lê Triệu Ngọc” và tên = “Đức”
- e) Thực hiện đảo ngược các từ có trong câu. Ví dụ: “cắm không được câu cá” → xuất ra: “cá câu được không cắm”
- f) Thực hiện kiểm tra chuỗi s có tuần hoàn với chu kỳ n hay không (n là số nguyên dương). Ví dụ: s = “abcabcabcabc” là chuỗi tuần hoàn với chu kỳ n = 3.

I. MỤC TIÊU:

Sau khi thực hành xong bài này, sinh viên có thể:

- Biết cách khai báo một mẫu tin (cấu trúc)
- Biết cách tạo mẫu tin có chứa các thành phần khác nhau (kiểu số, kiểu chuỗi..)
- Biết cách tạo danh sách các mẫu tin (mỗi phần tử trong danh sách là một mẫu tin)

II. TÓM TẮT LÝ THUYẾT:

Nội dung	Ví dụ
1. Cách khai báo mẫu tin	
<pre>struct TênCauTruc { <KieuDuLieu> tenthanhphan1; <Kieudulieu> tenthanhphan2; <Kieudulieu> tenthanhphan2; };</pre>	<pre>struct NhanVien { char maNhanVien[10]; char hoTen[50]; float luongCanBan; float heSoLuong; };</pre>
2. Cách khai báo danh sách mẫu tin	
<pre>struct TenCauTruc { <KieuDuLieu> tenthanhphan1; <KieuCauTruc> tendanhsach[MAX]; };</pre>	<pre>struct CongTy { int soNhanVien; NhanVien danhSachNhanVien[MAX]; };</pre>
3. Tìm lớn nhất (Luôn chọn được x_0 là phần tử đầu tiên thỏa điều kiện K)	<p>Chọn x_0 thuộc S sao cho thỏa điều kiện K và gán $Min = x_0$</p> <p>Duyệt x thuộc S (không chứa x_0)</p> <p style="padding-left: 40px;">Nếu x thỏa K và $Min > x$ thì</p> <p style="padding-left: 40px;">$Min = x$</p> <p>Cuối duyệt</p>
4. Tìm lớn nhất (Khó chọn được x_0 là phần tử đầu tiên thỏa điều kiện K (có thể có x_0 hoặc không có x_0))	<p>Flag = false</p> <p>Duyệt x thuộc S</p> <p style="padding-left: 40px;">Nếu x thỏa K</p> <p style="padding-left: 40px;">Nếu not Flag thì</p>

	Min = x; Flag = true Ngược lại nếu Min > x thì Min = x; Cuối nếu Cuối Duyệt
5. Kiểm tra tồn tại x thuộc tập S thỏa $P(x): (\exists x \in S) (P(x))$	KT := false Duyệt $x \in S$ Nếu P(x)thỏa thì KT := true Dừng quá trình duyệt S Cuối nếu Cuối duyệt

III. NỘI DUNG THỰC HÀNH:

1) Chương trình mẫu: (4đ)

Mô tả: Khai báo cấu trúc NhanVien, cấu trúc CongTy, các hàm nhập, xuất như trong 3 file Header.h, Source.cpp, Main.cpp.

Nội dung các hàm như sau:

1.1.Nhập thông tin 1 nhân viên

Dòng	Nội dung
1	<code>void nhapNhanVien(NhanVien &nhanVien)</code>
2	<code>{</code>
3	<code> fflush(stdin);</code>
4	<code> cout << "Ma nhan vien:";</code>
5	<code> cin.getline(nhanVien.maNhanVien, 10);</code>
6	<code> cout << "Ho ten:";</code>
7	<code> cin.getline(nhanVien.hoTen, 50);</code>
8	<code> cout << "Luong can ban:";</code>
9	<code> cin >> nhanVien.luongCanBan;</code>
10	<code> cout << "He so luong:";</code>
11	<code> cin >> nhanVien.heSoLuong;</code>
12	<code>}</code>

1.2. Xuất thông tin 1 nhân viên

Dòng	Nội dung
1	<code>void</code> xuấtNhanVien(<code>NhanVien</code> nv)
2	{
3	cout << "Ma nhan vien:" << nv.maNhanVien << endl;
4	cout << "Ho ten:" << nv.hoTen << endl;
5	cout << "Luong can ban:" << nv.luongCanBan << endl;
6	cout << "He so luong:" << nv.heSoLuong << endl;
7	}

1.3. Tính lương 1 nhân viên

Dòng	Nội dung
1	<code>float</code> tinhLuong(<code>NhanVien</code> nhanVien)
2	{
3	return nhanVien.luongCanBan*nhanVien.heSoLuong;
4	}

1.4. Nhập công ty (nhập danh sách nhân viên)

Dòng	Nội dung
1	<code>void</code> nhapCongTy(<code>CongTy</code> &cTy)
2	{
3	cout << "Nhap so luong nhan vien:";
4	cin >> cTy.soNhanVien;
5	cout << "Nhap danh sach nhan vien:" << endl;
6	for (<code>int</code> i = 0; i < cTy.soNhanVien; i++)
7	{
8	cout << "Nhap nhan vien thu " << i << endl;
9	nhapNhanVien(cTy.danhSach[i]);
10	}
11	}

1.5. Xuất công ty (xuất danh sách nhân viên)

Dòng	Nội dung
1	<code>void</code> xuấtCongTy(<code>CongTy</code> cTy)
2	{
3	cout << "\nSo luong nhan vien:" << cTy.soNhanVien << endl;
4	cout << "Danh sach nhan vien:" << endl;

5	<code>for (int i = 0; i < cTy.soNhanVien; i++)</code>
6	<code>{</code>
7	<code> xuatNhanVien(cTy.danhSach[i]);</code>
8	<code>}</code>
9	<code>}</code>

1.6. Tính tổng lương của công ty

Dòng	Nội dung
1	<code>float tinhTongLuong(CongTy cTy)</code>
2	<code>{</code>
3	<code> float tongLuong = 0;</code>
4	<code> for (int i = 0; i < cTy.soNhanVien; i++)</code>
5	<code> {</code>
6	<code> tongLuong = tongLuong + tinhLuong(cTy.danhSach[i]);</code>
7	<code> }</code>
8	<code> return tongLuong;</code>
9	<code>}</code>

1.7. Hàm tìm nhân viên theo mã:

Dòng	Nội dung
1	<code>bool timNhanVienTheoMa(CongTy cTy, NhanVien &nhanVien)</code>
2	<code>{</code>
3	<code> bool ketQua = false;</code>
4	<code> char maCanTim[10];</code>
5	
6	<code> fflush(stdin);</code>
7	<code> cout << "Nhap ma can tim:";</code>
8	<code> cin.getline(maCanTim, 10);</code>
9	
10	<code> for (int i = 0; i < cTy.soNhanVien; i++)</code>
11	<code> {</code>
12	<code> if (_stricmp(maCanTim, cTy.danhSach[i].maNhanVien) ==</code>
13	<code>0)</code>
14	<code> {</code>
15	<code> ketQua = true;</code>
16	<code> nhanVien = cTy.danhSach[i];</code>
17	<code> break;</code>

18	}
19	}
20	
21	return ketQua;
	}

1.8.Hàm main

Dòng	Nội dung
1	void hienThiMenu()
2	{
3	int chon = 0;
4	NhanVien nhanVien;
5	CongTy cTy;
6	CongTy cTyDaSapTangTheoHeSoLuong;
7	float tongLuong = 0;
8	float luongCaoNhat = 0;
9	CongTy dsNhanVienCoLuongCaoNhat;
10	bool ketQuaTimTheoMa = false;
11	bool ketQuaTimTheoTen = false;
12	
13	CongTy dsNhanVienTimTheoTen;
14	
15	do
16	{
17	cout << "\n0 - thoat chuong trinh:";
18	cout << "\n1 - nhap danh sach nhan vien:";
19	cout << "\n2 - xuat danh sach nhan vien:";
20	
21	cout << "\n3 - tinh tong luong:";
22	cout << "\n4 - tim luong cao nhat:";
23	cout << "\n5 - tim nhan vien co luong cao nhat:";
24	cout << "\n6 - tim nhan vien theo ma:";
25	cout << "\n7 - tim nhan vien theo ten:";
26	cout << "\nVui long chon:";
27	cin >> chon;
28	switch (chon)
29	{
30	case 0:
31	break;
32	case 1:

```

33         nhapCongTy(cTy);
34         break;
35     case 2:
36         xuấtCongTy(cTy);
37         break;
38     case 3:
39         tongLuong = tinhTongLuong(cTy);
40         cout << "Tong luong cua cong ty la:" << tongLuong
41 << endl;
42         break;
43     case 4:
44
45         break;
46     case 5:
47
48         break;
49     case 6:
50         ketQuaTimTheoMa = timNhanVienTheoMa(cTy,
51 nhanVien);
52         if (ketQuaTimTheoMa == true)
53         {
54             xuấtNhanVien(nhanVien);
55         }
56         else
57         {
58             cout << "Khong tim thay!";
59         }
60         break;
61     case 7:
62
63         break;
64     default:
65         cout << "Ban chon sai, vui long chon lai!";
66         break;
67     }
68 } while (chon != 0);
69
70 }
```

Yêu cầu:

A) Biên dịch chương trình.

B) Cho biết ý nghĩa của hàm `_strcmpi` tại dòng 12 trong phần 1.7

.....
.....
.....

C) Cho biết ý nghĩa của câu lệnh gán tại dòng 15 trong phần 1.7

.....
.....
.....

D) Cho biết ý nghĩa của câu lệnh `break` tại dòng 16 trong phần 1.7

.....
.....
.....

2) Chỉnh sửa số liệu theo yêu cầu: (2đ)

Viết hàm tìm lương cao nhất trong công ty.

Viết hàm tìm nhân viên có lương cao nhất đầu tiên

Viết hàm tìm nhân viên có lương cao nhất (tất cả nhân viên)

Viết hàm tìm nhân viên theo họ tên

3) Bài tập tổng hợp: (2đ)

Viết hàm tách tên nhân viên.

Viết hàm tìm nhân viên theo tên (theo tên chứ không phải theo họ tên)

I. MỤC TIÊU:

Sau khi thực hành xong bài này, sinh viên có thể:

- Thao tác trên file nhị phân (mở file, đọc/ghi, đóng file)
- Đọc/ghi danh sách mẫu tin trên file nhị phân.
- Biết thực hiện các phép toán trên danh sách (Danh sách mẫu tin) như: thêm, loại bỏ, tìm kiếm, sắp xếp, trộn, tách, ghép....

II. TÓM TẮT LÝ THUYẾT:

Nội dung	
1. Mở file nhị phân	
errno_t fopen_s(FILE **_File, const char* _Filename, const char* _Mode) _File: Con trỏ FILE _Filename: tên file chứa dữ liệu _Mode: chế độ mở file đọc/ghi “r”: chế độ đọc “w”: chế độ ghi Ví dụ: file_in = fopen_s(&file,tenFile, "r"); if (file == NULL) { cout << "Loi doc file!"; return; }	
2. Đóng file nhị phân	
int fclose(FILE * _File) Ví dụ: fclose(file);	
3. Đọc file nhị phân	
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream) ptr – Đây là con trỏ tới khối bộ nhớ có kích thước nhỏ nhất của khối size*nmemb bytes. size – Đây là kích thước (tính theo bytes) của mỗi phần tử được đọc. nmemb – Đây là số lượng các thành phần, mỗi thành phần có một kích thước tính theo bytes. stream – Đây là con trỏ đến đối tượng FILE đã mở trước đó.	
Ví dụ fread(&nhanVien, sizeof(NhanVien), 1, file);	
4. Ghi file nhị phân	
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)	

ptr – Đây là con trỏ đến mảng các thành phần được ghi
size - Đây là kích thước tính theo bytes của mỗi thành phần được ghi
nmemb – Đây là số các thành phần, mỗi thành phần có kích thước được tính theo bytes
stream – Đây là con trỏ đến đối tượng FILE đã mở trước đó.

Ví dụ:

```
fwrite(&cTy.danhSach[i], sizeof(NhanVien), 1, file);
```

5. Thêm phần tử vào cuối danh sách

Ý tưởng: Chỉ cần thêm vào cuối danh sách, sau đó tăng số lượng phần tử của danh sách hiện có thêm 1.

Ví dụ:

```
danhSach[kichThuoc] = X;  
kichThuoc++;
```

6. Loại bỏ phần tử tại vị trí cho trước

Ý tưởng:

B1: Bắt đầu từ vị trí cần xóa chạy đến phần tử gần cuối danh sách (kích thước danh sách -1)

B2: Gán phần tử phía sau cho phần tử trước đó

B3: Giảm kích thước danh sách xuống 1.

Ví dụ:

```
for(int i=viTriCanXoa; i< kichThuoc - 1; i++)  
{  
    danhSach[i] = danhSach[i+1];  
}  
kichThuoc--;
```

III. NỘI DUNG THỰC HÀNH:

1) Chương trình mẫu: (4đ)

Mô tả: Khai báo cấu trúc NhanVien, cấu trúc CongTy, các hàm nhập, xuất, đọc, ghi file như trong 3 file Header.h, Source.cpp, Main.cpp.

Nội dung các hàm như sau:

1.1. Nhập thông tin 1 nhân viên

Dòng	Nội dung
1	<code>void nhapNhanVien(NhanVien &nhanVien)</code>
2	<code>{</code>
3	<code> fflush(stdin);</code>
4	<code> cout << "Ma nhan vien:";</code>
5	<code> cin.getline(nhanVien.maNhanVien, 10);</code>
6	<code> cout << "Ho ten:";</code>
7	<code> cin.getline(nhanVien.hoTen, 50);</code>

8	cout << "Luong can ban:";
9	cin >> nhanVien.luongCanBan;
10	cout << "He so luong:";
11	cin >> nhanVien.heSoLuong;
12	}

1.2. Xuất thông tin 1 nhân viên

Dòng	Nội dung
1	void xuấtNhanVien(NhanVien nv)
2	{
3	cout << "Ma nhan vien:" << nv.maNhanVien << endl;
4	cout << "Ho ten:" << nv.hoTen << endl;
5	cout << "Luong can ban:" << nv.luongCanBan << endl;
6	cout << "He so luong:" << nv.heSoLuong << endl;
7	}

1.3. Nhập công ty (nhập danh sách nhân viên)

Dòng	Nội dung
1	void nhapCongTy(CongTy &cTy)
2	{
3	cout << "Nhap so luong nhan vien:";
4	cin >> cTy.soNhanVien;
5	cout << "Nhap danh sach nhan vien:" << endl;
6	for (int i = 0; i < cTy.soNhanVien; i++)
7	{
8	cout << "Nhap nhan vien thu " << i << endl;
9	nhapNhanVien(cTy.danhSach[i]);
10	}
11	}

1.4. Xuất công ty (xuất danh sách nhân viên)

Dòng	Nội dung
1	<code>void</code> xuấtCongTy(<code>CongTy</code> cTy)
2	{
3	cout << "\nSo luong nhan vien:" << cTy.soNhanVien << endl;
4	cout << "Danh sach nhan vien:" << endl;
5	for (<code>int</code> i = 0; i < cTy.soNhanVien; i++)
6	{
7	xuấtNhanVien(cTy.danhSach[i]);
8	}
9	}

1.5. Đọc danh sách nhân viên của công ty từ file

Dòng	Nội dung
1	<code>void</code> docFile(<code>const char*</code> tenFile, <code>CongTy</code> &cTy)
2	{
3	FILE *file;
4	errno_t file_in;
5	//mở file để đọc
6	file_in = fopen_s(&file, tenFile, "r");
7	if (file == NULL)
8	{
9	cout << "Lỗi đọc file!";
10	return;
11	}
12	int i = 0;
13	NhanVien nhanVien;
14	// đọc nội dung file đến khi kết thúc file
15	while (fread(&nhanVien, sizeof(NhanVien), 1, file))
16	{
17	cTy.danhSach[i++] = nhanVien;
18	}
19	cTy.soNhanVien = i;
20	// đóng file
21	fclose(file);
22	}

1.6. Ghi danh sách nhân viên của công ty vào file:

Dòng	Nội dung
------	----------

```
1 void ghiFile(const char* tenFile, CongTy cTy)
2 {
3     FILE *file;
4     errno_t file_out;
5     // mo file de ghi
6     file_out = fopen_s(&file, tenFile, "w");
7     if (file == NULL)
8     {
9         cout<<"\nLoi mo file\n";
10        return;
11    }
12    //ghi từng dòng
13    for (int i = 0; i < cTy.soNhanVien; i++)
14    {
15        fwrite(&cTy.danhSach[i], sizeof(NhanVien), 1, file);
16    }
17    if (fwrite == 0)
18    {
19        cout<<"\nLoi ghi file\n";
20        return;
21    }
22    else{
23        cout << "Ghi file thanh cong!";
24    }
25    //dong file
26    fclose(file);
27 }
```

1.7. Thêm nhân viên

Dòng	Nội dung
1	<code>void themNhanVien(CongTy &cTy)</code>
2	<code>{</code>
3	<code> NhanVien nhanVien;</code>
4	<code> cout << "\nNhap thong tin nhan vien can them:";</code>
5	<code> nhapNhanVien(nhanVien);</code>
6	<code> cTy.danhSach[cTy.soNhanVien++] = nhanVien;</code>
7	<code>}</code>

1.8. Xóa nhân viên

Dòng	Nội dung
1	<code>bool xoaNhanVien(CongTy &cTy)</code>
2	<code>{</code>
3	<code> bool ketQua = false;</code>


```
4      char maCanXoa[10];
5      fflush(stdin);
6      cout << "Nhap ma nhan vien can xoa:";
7      cin.getline(maCanXoa, 10);
8      //tim vi tri can xoa
9      int viTri = -1;
10     for (int i = 0; i < cTy.soNhanVien; i++)
11     {
12         if (_strcmapi(maCanXoa, cTy.danhSach[i].maNhanVien)
13 == 0)
14         {
15             viTri = i;
16             break;
17         }
18     }
19     //xoa
20     if (viTri != -1)
21     {
22         for (int i = viTri; i < cTy.soNhanVien - 1; i++)
23         {
24             cTy.danhSach[i] = cTy.danhSach[i + 1];
25         }
26         cTy.soNhanVien--;
27         ketQua = true;
28     }
29     return ketQua;
30 }
```

1.9.Sắp xếp nhân viên tăng dần theo hệ số lương

Dòng	Nội dung
1	<code>CongTy sapXepNhanVienTangDanTheoHeSoLuong(CongTy cTy)</code>
2	{
3	<code>CongTy cTyKetQua;</code>
4	<code>cTyKetQua = cTy;</code>
5	for (int i = 0; i < cTyKetQua.soNhanVien; i++)
6	{
7	for (int j = i + 1; j < cTyKetQua.soNhanVien; j++)
8	{
9	if (cTyKetQua.danhSach[i].heSoLuong >
10	cTyKetQua.danhSach[j].heSoLuong)
11	{
12	//hoan vi nhan vien thu i va j

13	<code>NhanVien nhanVienTam;</code>
14	<code>nhanVienTam = cTyKetQua.danhSach[i];</code>
15	<code>cTyKetQua.danhSach[i] =</code>
16	<code>cTyKetQua.danhSach[j];</code>
17	<code>cTyKetQua.danhSach[j] = nhanVienTam;</code>
18	<code>}</code>
19	<code>}</code>
20	<code>}</code>
21	
22	<code>return cTyKetQua;</code>
23	<code>}</code>

1.10. Hàm main

Dòng	Nội dung
1	<code>void hienThiMenu()</code>
2	<code>{</code>
3	<code>int chon = 0;</code>
4	<code>NhanVien nhanVien;</code>
5	<code>CongTy cTy;</code>
6	<code>CongTy cTyDaSapTangTheoHeSoLuong;</code>
7	<code>do</code>
8	<code>{</code>
9	<code>cout << "\n0 - thoat chuong trinh:";</code>
10	<code>cout << "\n1 - nhap danh sach nhan vien:";</code>
11	<code>cout << "\n2 - xuat danh sach nhan vien:";</code>
12	
13	<code>cout << "\n3 - doc file:";</code>
14	<code>cout << "\n4 - ghi file:";</code>
15	<code>cout << "\n5 - them nhan vien:";</code>
16	<code>cout << "\n6 - xoa nhan vien:";</code>
17	<code>cout << "\n7 - sap xep tang dan theo he so luong:";</code>
18	<code>cout << "\nVui long chon:";</code>
19	<code>cin >> chon;</code>
20	<code>switch (chon)</code>
21	<code>{</code>
22	<code>case 0:</code>
23	<code>break;</code>
24	<code>case 1:</code>
25	<code>nhapCongTy(cTy);</code>
26	<code>break;</code>
27	<code>case 2:</code>
28	<code>xuatCongTy(cTy);</code>

```
29         break;
30     case 3:
31         docFile("danhsach.dat", cTy);
32         break;
33     case 4:
34         ghiFile("danhsach.dat", cTy);
35         break;
36     case 5:
37         themNhanVien(cTy);
38         break;
39     case 6:
40         xoaNhanVien(cTy);
41         break;
42     case 7:
43         cTyDaSapTangTheoHeSoLuong =
44     sapXepNhanVienTangDanTheoHeSoLuong(cTy);
45         xuấtCongTy(cTyDaSapTangTheoHeSoLuong);
46         break;
47     default:
48         cout << "Ban chon sai, vui long chon lai!";
49         break;
50     }
51     } while (chon != 0);
52
53 }
```

Yêu cầu:

E) Biên dịch chương trình.

F) Cho biết ý nghĩa của câu lệnh tại dòng 15 phần 1.8

.....
.....
.....

G) Cho biết ý nghĩa của câu lệnh tại dòng 16 trong phần 1.8

.....
.....
.....

H) Cho biết ý nghĩa của câu lệnh tại dòng 20 trong phần 1.8

.....
.....
.....

2) Chỉnh sửa số liệu theo yêu cầu: (2đ)

- Viết hàm thêm nhân viên vào 1 vị trí bất kỳ trong danh sách.
- Viết hàm tìm nhân viên theo hệ số lương (phải trả về danh sách nhân viên)
- Viết hàm tách danh sách nhân viên theo hệ số lương cho trước (1 danh sách là nhân viên có hệ số nhỏ hơn hệ số lương cho trước và 1 danh sách là nhân viên có hệ số lớn hơn hệ số lương cho trước).
- Viết hàm trộn 2 danh sách nhân viên cho trước thành 1 danh sách.

3) Bài tập tổng hợp: (2đ)

- Viết hàm hoán vị 2 nhân viên; sau đó sửa lại hàm sắp xếp trong phần 1.9 để gọi hàm hoán vị này.
- Viết hàm trộn 2 danh sách nhân viên sao cho kết quả là 1 danh sách tăng dần theo hệ số lương.

I. MỤC TIÊU:

Sau khi thực hành xong bài này, sinh viên có thể :

- Biết cách lập trình đệ quy.
- Ứng dụng đệ quy cài đặt thuật toán QuickSort và MergeSort.
- Cài đặt thuật toán tìm kiếm nhị phân trên dãy số đã được sắp xếp.

II. TÓM TẮT LÝ THUYẾT:**1) Đệ quy:**

- Đệ quy (Recursion) là một trong những giải thuật khá quen thuộc trong lập trình mở rộng ra trong toán học (thường được gọi là quy nạp)

VD: $n! = n * (n - 1)!$ ($n \geq 0$)

- Một đối tượng được mô tả thông qua chính nó được gọi là mô tả đệ quy.

VD: Một node trong cây nhị phân chứa thông tin của node và hai node con của nó cũng là một node.

```
struct node {  
    int info;  
    node* pLeft;  
    node* pRight;  
};
```

- **Hàm đệ quy:**

Trong lập trình, một hàm được gọi là đệ quy khi nó gọi chính nó trong thân hàm.

VD:

```
void HamDeQuy() {  
    HamDeQuy();  
}
```

Hàm đệ quy gồm 2 phần:

- Phần cơ sở: Điều kiện thoát khỏi đệ quy.
- Phần đệ quy: Thân hàm có chứa lời gọi đệ quy.
- **Thiết kế giải thuật đệ quy:**
 - Xác định tham số bài toán.

- Phân tích trường hợp chung: Đưa bài toán về bài toán nhỏ hơn cùng loại, dần dần tiến tới trường hợp suy biến.
- Tìm trường hợp suy biến (điều kiện thoát khỏi đệ quy).

2) Merge Sort (Sắp xếp trộn):

Merge sort là một thuật toán chia để trị. Thuật toán này chia mảng cần sắp xếp thành 2 nửa. Tiếp tục lặp lại việc này ở các nửa mảng đã chia. Sau cùng gộp các nửa đó thành mảng đã sắp xếp. Hàm merge() được sử dụng để gộp hai nửa mảng. Hàm merge(arr, l, m, r) là tiến trình quan trọng nhất sẽ gộp hai nửa mảng thành 1 mảng sắp xếp, các nửa mảng là arr[l...m] và arr[m+1...r] sau khi gộp sẽ thành một mảng duy nhất đã sắp xếp.

▪ Thuật toán merge sort:

MergeSort(a [], l, r)

Nếu $r > l$

1. Tìm chỉ số nằm giữa mảng để chia mảng thành 2 nửa:

$$m = (l+r)/2$$

2. Gọi đệ quy hàm MergeSort cho nửa đầu tiên:

MergeSort(a, l, m)

3. Gọi đệ quy hàm mergeSort cho nửa thứ hai:

MergeSort(a, m+1, r)

4. Gộp 2 nửa mảng đã sắp xếp ở (2) và (3):

Merge(a, l, m, r)

Merge(a[], l, m, r)

1. Chép các phần tử ra các mảng tạm (mảng L từ l đến m, mảng R từ m+1 đến r)
2. Trộn hai mảng tạm vào lại mảng a sao cho các phần tử đứng đúng thứ tự.
3. Chép các phần tử còn lại sau khi trộn (nếu có) của hai mảng.

3) Quick Sort (Sắp xếp nhanh):

Thuật toán sắp xếp quick sort là một thuật toán chia để trị. Nó chọn một phần tử trong mảng làm điểm đánh dấu (pivot). Thuật toán sẽ thực hiện chia mảng thành các mảng con dựa vào pivot đã chọn. Việc lựa chọn pivot ảnh hưởng rất nhiều tới tốc độ sắp xếp. Dưới đây là một số cách để chọn pivot thường được sử dụng:

1. Luôn chọn phần tử đầu tiên của mảng.
2. Luôn chọn phần tử cuối cùng của mảng. (*Được sử dụng trong bài lab này*)
3. Chọn một phần tử ngẫu nhiên.
4. Chọn một phần tử có giá trị nằm giữa mảng.

Mấu chốt chính của thuật toán quick sort là việc phân hoạch dãy số (Xem hàm partition()). Cho một mảng và một phần tử x là pivot. Đặt x vào đúng vị trí của mảng đã sắp xếp. Di chuyển tất cả các phần tử của mảng nhỏ hơn x sang bên trái vị trí của x , và di chuyển tất cả các phần tử của mảng mà lớn hơn x sang bên phải vị trí của x .

▪ **Thuật toán phân hoạch trong quick sort:**

- Đặt pivot là phần tử cuối cùng của dãy số a .
- Đặt phần tử trái nhất của dãy số có chỉ số là $left$, phần tử phải nhất của dãy số có chỉ số là $right - 1$ (bỏ qua phần tử pivot).
- Trong khi $left < right$ mà $a[left] > pivot$ và $a[right] < pivot$ thì đổi chỗ hai phần tử $left$ và $right$.
- Sau cùng, ta đổi chỗ hai phần tử $left$ và pivot cho nhau. Khi đó, phần tử $left$ đã đứng đúng vị trí và chia dãy số làm hai (bên trái và bên phải)

4) Tìm kiếm nhị phân (Binary Search)

Cho mảng a đã sắp xếp tăng dần, việc tìm kiếm phần tử x có thể thực hiện như sau:

1. Xét đoạn mảng $a[left...right]$ cần tìm kiếm phần tử x . Ta so sánh x với phần tử ở vị trí giữa của mảng ($mid = (left + right)/2$).
2. Nếu phần tử $a[mid] = x$. Kết luận và thoát chương trình.
3. Nếu $a[mid] < x$. Chỉ thực hiện tìm kiếm trên đoạn $arr[mid+1...right]$.
4. Nếu $a[mid] > x$. Chỉ thực hiện tìm kiếm trên đoạn $arr[left...mid-1]$

III. NỘI DUNG THỰC HÀNH:

1/ Chương trình mẫu: (4 điểm)

Tạo 3 file chương trình mẫu: **thuvien_lab9.h**, **caidat_lab9.cpp**, **chuongtrinh_lab9.cpp** có nội dung như sau:

```
thuvien_lab9.h  X
Lab9
1  #include <iostream>
2  using namespace std;
3  const int MAX = 50;
4  struct DAYSO {
5      int n;
6      int list[MAX];
7  };
8  int Fibonacci(int n);
9  int GiaiThua(int n);
10 int BinarySearch(int a[], int l, int r, int x);
11 void XuatDAYSO(DAYSO a);
12 void MergeSort(int a[], int l, int r);
13 void merge(int a[], int l, int m, int r);
14 void QuickSort(int a[], int l, int h);
15 int partition(int a[], int low, int high);
16 void hoanvi(int &a, int &b);
17 //Bài tập tổng hợp
18 struct SinhVien {
19     char mssv[5];
20     char holot[30];
21     char ten[10];
22     float dtb;
23 };
24 struct DSSV {
25     int siso;
26     SinhVien list[MAX];
27 };
28 DSSV TaoDuLieu();
29 void InSinhVien(SinhVien sv);
30 void InDSSV(DSSV dssv);
31 void MergeSort(SinhVien a[], int l, int r);
32 void QuickSort(SinhVien a[], int l, int h);
33 int BinarySearch(SinhVien a[], int l, int r, char* mssv);
```



```
caidat_lab9.cpp X
Lab9
1 #include "thuvien_lab9.h"
2
3 void XuatDAYSO(DAYSO a) {
4     cout << endl;
5     for (int i = 0; i < a.n; i++)
6         cout << " " << a.list[i];
7 }
8
9 void hoanvi(int &a, int &b) {
10     int t = a;
11     a = b;
12     b = t;
13 }
14
15 int Fibonaci(int n) {
16     if (n < 3) return 1;
17     return Fibonaci(n - 1) + Fibonaci(n - 2);
18 }
19 int GiaiThua(int n) {
20     if (n == 0 || n == 1) return 1;
21     return n * GiaiThua(n - 1);
22 }
23
24 // Hàm tìm kiếm nhị phân sử dụng giải thuật đệ quy
25 int BinarySearch(int a[], int l, int r, int x) {
26     if (r >= l) {
27         int mid = (l + r) / 2;
28
29         // Nếu a[mid] = x, trả về chỉ số và kết thúc.
30         if (a[mid] == x)
31             return mid;
32
33         // Nếu a[mid] > x, thực hiện tìm kiếm nửa trái của mảng
34         if (a[mid] > x)
35             return BinarySearch(a, l, mid - 1, x);
36         // Nếu a[mid] < x, thực hiện tìm kiếm nửa phải của mảng
37         return BinarySearch(a, mid + 1, r, x);
38     }
39
40     // Nếu không tìm thấy
41     return -1;
42 }
43 //*****
44
45 //Thuật toán MergeSort
46
47 /* l là chỉ số trái và r là chỉ số phải của mảng cần được sắp xếp */
48 void MergeSort(int a[], int l, int r) {
49     if (l < r) {
50         int m = (l+r) / 2;
51
```

```

52         // Gọi hàm đệ quy tiếp tục chia đôi từng nửa mảng
53         MergeSort(a, l, m);
54         MergeSort(a, m + 1, r);
55
56         merge(a, l, m, r);
57     }
58 }
59
60 // Gộp hai mảng con a[l...m] và a[m+1..r]
61 void merge(int a[], int l, int m, int r) {
62     int i, j, k;
63     int n1 = m - l + 1;
64     int n2 = r - m;
65
66     /* Tạo các mảng tạm */
67     int *L, *R;
68     L = new int[n1];
69     R = new int[n2];
70
71     /* Copy dữ liệu sang các mảng tạm */
72     for (i = 0; i < n1; i++)
73         L[i] = a[l + i];
74     for (j = 0; j < n2; j++)
75         R[j] = a[m + 1 + j];
76
77     /* Gộp hai mảng tạm vừa rồi vào mảng a */
78     i = 0; // Khởi tạo chỉ số bắt đầu của mảng con đầu tiên
79     j = 0; // Khởi tạo chỉ số bắt đầu của mảng con thứ hai
80     k = l; // Khởi tạo chỉ số bắt đầu của mảng lưu kết quả
81     while (i < n1 && j < n2) {
82         if (L[i] <= R[j]) {
83             a[k] = L[i];
84             i++;
85         }
86         else {
87             a[k] = R[j];
88             j++;
89         }
90         k++;
91     }
92     /* Copy các phần tử còn lại của mảng L vào a nếu có */
93     while (i < n1) {
94         a[k] = L[i];
95         i++;
96         k++;
97     }
98     /* Copy các phần tử còn lại của mảng R vào a nếu có */
99     while (j < n2) {
100         a[k] = R[j];
101         j++;
102         k++;
103     }
104
105     delete L;

```

```
106     delete R;
107 }
108
109 //*****
110
111 // Thuật toán QuickSort
112
113 void QuickSort(int a[], int low, int high) {
114     if (low < high) {
115         /* pi là chỉ số nơi phần tử này đã đứng đúng vị trí
116         và là phần tử chia mảng làm 2 mảng con trái & phải */
117         int pi = partition(a, low, high);
118
119         // Gọi đệ quy sắp xếp 2 mảng con trái và phải
120         QuickSort(a, low, pi - 1);
121         QuickSort(a, pi + 1, high);
122     }
123 }
124
125 int partition(int a[], int low, int high) {
126     int pivot = a[high];    // pivot
127     int left = low;
128     int right = high - 1;
129     while (1) {
130         while (left <= right && a[left] < pivot) left++;
131         while (right >= left && a[right] > pivot) right--;
132         if (left >= right) break;
133         hoanvi(a[left], a[right]);
134         left++;
135         right--;
136     }
137     hoanvi(a[left], a[high]);
138     return left;
139 }
140
141 //*****
142
143 //Bài tập tổng hợp
144 DSSV TaoDuLieu() {
145     DSSV dssv = { 10, {
146         { "SV01", "Le Thi", "Be", 7.7 },
147         { "SV10", "Tran Van", "An", 6.5 },
148         { "SV03", "Nguyen Ngoc", "Ti", 4.5 },
149         { "SV06", "Le Van", "Phuoc", 8.0 },
150         { "SV04", "Tran Thi", "Binh", 6.0 },
151         { "SV05", "Ngo Bao", "Chau", 7.0 },
152         { "SV07", "Le Thanh", "Tam", 9.0 },
153         { "SV09", "Mac Thi", "Bui", 10.0 },
154         { "SV02", "Le Thi Hong", "Gam", 3.5 },
155         { "SV08", "Nguyen Duy", "An", 8.5 }
156     }
157 };
158     return dssv;
159 }
```

```
160 void InSinhVien(SinhVien sv) {
161     cout << endl << sv.mssv << "\t" << sv.hoLot << " " << sv.ten << "\t" << sv.dtb;
162 }
163 void InDSSV(DSSV dssv) {
164     cout << endl;
165     for (int i = 0; i < dssv.siso; i++) {
166         InSinhVien(dssv.list[i]);
167         cout << endl;
168     }
169 }
170 void MergeSort(SinhVien a[], int l, int r) {
171 }
172 }
173 void QuickSort(SinhVien a[], int l, int h) {
174 }
175 }
176 int BinarySearch(SinhVien a[], int l, int r, char* mssv) {
177     return -1;
178 }
179 }
180 //*****
```

Mô tả chương trình:

- Nhập n nguyên dương. In ra n giai thừa. In ra phần tử thứ n của dãy Fibonacci.
- Sắp thứ tự dãy A bằng thuật toán MergeSort. In ra dãy A đã sắp tăng dần.
- Sắp thứ tự dãy B bằng thuật toán QuickSort. In ra dãy B đã sắp tăng dần.
- Nhập giá x cần tìm trong dãy A . Dùng thuật toán Tìm kiếm nhị phân tìm vị trí của x có trong dãy A .

Yêu cầu sinh viên:

- Biên dịch chương trình trên.
- Cho biết kết quả in ra màn hình khi người dùng nhập dữ liệu sau:
 $n = 5, x = 8$
 $n = 7, x = 10$
- Chạy từng bước hàm **GiaiThua** với $n = 5$.

.....

.....

.....

.....

.....

d) Chạy từng bước hàm **Fibonacci** với $n = 7$

e) Cho mảng: $a = \{ 4, 23, 5, 8, 7, 9, 12, 16, 11, 17 \}$. Chạy từng bước hàm **merge(a, 0, 2, 5)** \rightarrow cho biết các giá trị $L[i]$, $R[j]$, $a[k]$ qua mỗi bước lặp.

- f) Cho mảng: $b = \{ 2, 9, 5, 15, 41, 32, 7, 3, 8, 17 \}$. Chạy từng bước hàm **partition(a, 0, 9)** → cho biết các giá trị left, right khi hoán vị hai phần tử $a[\text{left}], a[\text{right}]$, kết quả mảng a sau khi thực thi hàm là gì.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2/ Chính sửa số liệu theo yêu cầu: (0.2 điểm)

- a) Thay đổi thuật toán Merge Sort sắp xếp mảng giảm dần.
- b) Thay đổi thuật toán Quick Sort sắp xếp mảng giảm dần.
- c) Thay đổi thuật toán Binary Search tìm kiếm trên mảng có thứ tự giảm dần.

3/ Bài tập tổng hợp: (0.2 điểm)

- a) Cài đặt thuật toán Merge Sort để sắp danh sách sinh viên tăng dần theo MSSV.
- b) Cài đặt thuật toán tìm kiếm nhị phân theo MSSV.
- c) Cài đặt thuật toán Quick Sort để sắp danh sách sinh viên tăng dần theo tên, nếu trùng tên thì sắp tăng dần theo họ lót.

4/ Bài tập làm thêm: (0.2 điểm)

Dùng đệ qui để giải các bài toán sau:

- a) Tìm chữ số lớn nhất của số nguyên dương n .
- b) Tìm chữ số nhỏ nhất của số nguyên dương n .
- c) Kiểm tra số nguyên dương n có toàn chữ số chẵn hay không?
- d) Kiểm tra số nguyên dương n có toàn chữ số lẻ hay không?
- e) Cho số nguyên dương n , tìm chữ số đầu tiên của n .