CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

A. BÀI TOÁN TÌM KIẾM:

1. Các thuật toán cần ghi nhớ:

a) Thuật toán tìm kiếm tuyến tính:

```
int Search(int a[],int n,int key)
{
	for (int i=0;i<n;i++)
		if (a[i]==key) return i; //tìm thấy tại vị trí i
	return -1; //không tìm thấy
}
```

➔ Đối với SV: so sánh mssv => **strcmp(a[i].mssv,a[j].mssv)**

b) Thuật toán tìm kiếm nhị phân:

```
int BinarySearch(int a[],int n,int key)
{
	int left=0,right=n-1,mid; //tìm kiếm tất cả phần tử
	while (left<=right)
	{
		mid=(left+right)/2; //lấy điểm giữa
		if (a[mid]==key) return mid; //nếu tìm được
		if (a[mid]<key) left=mid+1; //tìm đoạn bên phải mid
		else right=mid-1; //tìm đoạn bên trái mid
	}
	return -1; //không tìm được
}
```

➔ Đối với SV: so sánh mssv => **strcmp(a[i].mssv,a[j].mssv)**

2. Các bước để làm bài toán tìm kiếm:

- Bước 1: Khai báo các thư viện cần thiết:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h> //nếu dùng hàm fflush(stdin);
```

- Bước 2: Khai báo macro tráo đổi giá trị 2 biến a, b có kiểu là type (nếu có dùng sắp xếp), và số phần tử tối đa:

```
#define Swap(type,a,b) {type tmp=a; a=b; b=tmp;} //không có ; ở cuối
#define MAX_ARR 100 //không có ; ở cuối
```

➔ Đối với SV: Swap(SV,?,?)

- Bước 3: Nếu bài toán là quản lí sinh viên, sách, hàng hoá,… thì cần khai báo cấu trúc cho loại đó; và viết hàm nhập xuất 1 sinh viên, hàng hoá,…; và nhiều sinh viên, hàng hoá,…

```
struct sinhvien
{
	char mssv[11];
```

```
        char ten[31];
        float dtb;
}; //nhớ có dấu ; nhé
typedef struct sinhvien SV; //nhớ có hàng này
void Nhap1SV(SV &x)
{
        printf("\t-Nhap mssv: ");
        fflush(stdin);
        gets(x.mssv);
        printf("\t-Nhap ten: ");
        fflush(stdin);
        gets(x.ten);
        float tmp; //nhập cho trường kiểu số thực phải dùng biến tạm
        printf("\t-Nhap diem: ");
        scanf("%f",&tmp);
        x.diem=tmp;
}
void Xuat1SV(SV x)
{
        printf("%s\t%s\t%.2f",x.mssv,x.ten,x.diem);
}
void NhapSV(SV a[],int n)
{
        for (int i=0;i<n;i++)
        {
                printf("Nhap thong tin SV thu %d:\n",i+1);
                Nhap1SV(a[i]);
                printf("\n---------\n");
        }
}
void XuatSV(SV a[],int n)
{
        printf("MSSV\tTEN\tDIEM\n");
        for (int i=0;i<n;i++)
        {
                Xuat1SV(a[i]);
                printf("\n");
        }
}
```

- Bước 4: Viết hàm nhập xuất mảng (nếu đề yêu cầu thực hiện trên mảng):

```
void Nhapmang(int a[],int n)
{
        for (int i=0;i<n;i++)
        {
```

```
            printf("Nhap phan tu a[%d] = ",i+1);
            scanf("%d",&a[i]);
        }
}
void Xuatmang(int a[],int n)
{
        for (int i=0;i<n;i++)
            printf("%d\t",a[i]);
}
```

- Bước 5: Nhập số lượng (phần tử, sinh viên, hàng hoá,…):

```
void NhapN(int &n)
{
        do
        {
            printf("Nhap so luong N = ");
            scanf("%d",&n);
            if (n<1 || n>MAX_ARR) printf("Ban nhap sai. Moi nhap lai.\n");
        }
        while (n<1 || n>MAX_ARR); //nhớ phải kiểm tra điều kiện nhé
}
```

- Bước 6: Nếu dùng thuật toán tìm kiếm nhị phân, phải có hàm sắp xếp, sử dụng thuật toán sắp xếp nổi bọt:

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

➔ Đối với SV: so sánh mssv => **strcmp(a[i].mssv,a[j].mssv)**

- Bước 7: Viết thuật toán tìm kiếm cần sử dụng.
- Bước 8: Chương trình chính:

```
int main() //bắt đầu chương trình chính
{
        clrscr(); //xoá màn hình
        int n; int a[MAX_ARR]; //nếu SV thì khai báo: SV a[MAX_ARR];
        NhapN(n); //nhập giá trị cho N
        Nhapmang(a,n); //nếu là SV thì: NhapSV(a,n);
        printf("Mang vua nhap:\n");
        Xuatmang(a,n); //nếu là SV thì: XuatSV(a,n);
        BubbleSort(a,n); //nếu dùng thuật toán tìm kiếm nhị phân
        //nhập giá trị cần tìm
        int k; //nếu là SV thì dựa vào kiểu của các trường như char, float,…
        printf("\nNhap gia tri can tim: ");
        scanf("%d",&k);
```

```
        //thực thi thuật toán tìm kiếm tuyến tính
        if (Search(a,n,k)!==-1) //không tìm thấy
                printf("Khong tim thay!\n");
        else printf("Tim thay tai vi tri: %d",Search(a,n,k));
        //thực thi thuật toán tìm kiếm nhị phân tương tự.
        getch(); //dừng màn hình
        return 0; //mã thoát hàm main
}
```

HƯỚNG DẪN CHI TIẾT ĐẾN ĐÂY MÀ CÁC BẠN VẪN KHÔNG LÀM ĐƯỢC NỮA THÌ "CHÚNG TA KHÔNG THUỘC VỀ NHAU" NHÉ ☺

B. BÀI TOÁN SẮP XẾP

1. Các thuật toán cần ghi nhớ:

➔ Đối với kiểu sinh viên, khi so sánh mssv, tensv,… (chuỗi) ta dùng hàm strcmp(a,b), < 0 thì sắp xếp tăng dần, > 0 thì sắp xếp giảm dần. Ví dụ: **strcmp(a[i].mssv,a[j].mssv)**

a) Thuật toán sắp xếp nổi bọt (BubbleSort):

```
void BubbleSort(int a[],int n)
{
        for (int i=0;i<n-1;i++)
                for (int j=n-1;j>i;j--) //đổi chỗ cặp phần tử đứng sai
                        if (a[j-1]>a[j]) //phần tử đứng trước > phần tử đứng sau
                                Swap(int,a[j],a[j-1]);
}
```

b) Thuật toán sắp xếp đổi chỗ trực tiếp (InterchangeSort):

```
void InterchangeSort(int a[],int n)
{
        for (int i=0;i<n-1;i++)
                for (int j=i+1;j<n;j++) //duyệt các phần tử đứng sau
                        if (a[j]<a[i]) //phần tử đứng sau < phần tử đứng trước
                                Swap(int,a[i],a[j]);
}
```

c) Thuật toán sắp xếp chọn trực tiếp (SelectionSort):

```
void SelectionSort(int a[],int n)
{
        for (int i=0;i<n-1;i++) //duyệt qua n phần tử
        {
                int minpos=i; //lưu vị trí i hiện tại
                for (int j=i+1;j<n;j++) //duyệt các phần tử phía sau
                        if (a[j]<a[minpos]) minpos=j; //phần tử sau < phần tử trước
                Swap(int,a[minpos],a[i]);
        }
}
```

d) Thuật toán sắp xếp chèn trực tiếp (InsertionSort):

```
void InsertionSort(int a[],int n)
{
	int pos; int x; //x lưu phần tử a[i]
	for (int i=1;i<n;i++)
	{
		x=a[i]; pos=i-1; //xét từ vị trí i trở về trước
		while (pos>=0 && a[pos]>x)
		//dời chỗ các phần tử đứng sau x trong dãy mới
		{
			a[pos+1]=a[pos];
			pos--;
		}
		a[pos+1]=x;
	}
}
```

e) Thuật toán ShellSort:

```
void ShellSort(int a[],int n)
{
	int gap,i,j;
	for (gap=n/2;gap>0;gap/=2)
		for (i=gap;i<n;i++) //xét từ gap trở về sau
			for (j=i-gap;j>=0 && a[j]>a[j+gap];j-=gap) //kiểm tra
				Swap(int,a[j],a[j+gap]);
}
```

f) Thuật toán sắp xếp kiểu phân đoạn (QuickSort):

```
void QuickSort(int a[],int left,int right)
{
	int i,j; int x;
	x=a[(left+right)/2]; //chọn phần tử giữa làm mốc
	i=left; j=right;
	do
	{
		while (a[i]<x) i++; //lặp đến khi a[i]>=x
		while (a[j]>x) j--; //lặp đến khi a[j]<=x
		if (i<=j)
		{
			Swap(int,a[i],a[j]);
			i++; //qua phần tử kế tiếp
			j--; //qua phần tử đứng trước
		}
	}
	while (i<j);
	if (left<j) QuickSort(a,left,j); //phân đoạn bên trái
	if (right>i) QuickSort(a,i,right); //phân đoạn bên phải
```

```
}
```

g) Thuật toán sắp xếp trộn (MergeSort):

```
void MergeSort(int a[],int n)
{
	int i,j,k,low1,up1,low2,up2,size;
	int tmp[MAX_ARR]; size=1;
	while (size<n)
	{
		low1=0; k=0;
		while (low1+size<n)
		{
			low2=low1+size;
			up1=low2-1;
			up2=(low2+size-1<n) ? (low2+size-1):(n-1);
			for (i=low1,j=low2;i<=up1 && j<=up2;k++)
				if (a[i]<=a[j]) tmp[k]=a[i++];
			else tmp[k]=a[j++];
			for (;i<=up1;k++) tmp[k]=a[i++];
			for (;j<=up2;k++) tmp[k]=a[j++];
			low1=up2+1;
		}
		for (i=low1;k<n;i++) tmp[k++]=a[i];
		for (i=0;i<n;i++) a[i]=tmp[i];
		size*=2;
	}
}
```

h) Thuật toán sắp xếp chèn nhị phân:

```
void BInsertionSort(int a[],int n)
{
	int l,r,m; int x;
	for (int 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(int j=i-1;j>=l;j--) a[j+1]=a[j];
		a[l]=x;
	}
}
```

➔ ***Nhấn mạnh:*** InterchangeSort, BubbleSort, SelectionSort, InsertionSort, QuickSort.

2. Các bước để làm một bài toán sắp xếp:

- Bước 1: Khai báo các thư viện:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h> //nếu dùng hàm fflush(stdin);
```

- Bước 2: Khai báo macro tráo đổi giá trị 2 biến a, b có kiểu là type (nếu có dùng sắp xếp), và số phần tử tối đa:

```
#define Swap(type,a,b) {type tmp=a; a=b; b=tmp;} //không có ; ở cuối
#define MAX_ARR 100 //không có ; ở cuối
```

➔ Đối với SV: Swap(SV,?,?)

- Bước 3: Nếu bài toán là quản lí sinh viên, sách, hàng hoá,… thì cần khai báo cấu trúc cho loại đó; và viết hàm nhập xuất 1 sinh viên, hàng hoá,…; và nhiều sinh viên, hàng hoá,…

```
struct sinhvien
{
        char mssv[11];
        char ten[31];
        float dtb;
}; //nhớ có dấu ; nhé
typedef struct sinhvien SV; //nhớ có hàng này
void Nhap1SV(SV &x)
{
        printf("\t-Nhap mssv: ");
        fflush(stdin);
        gets(x.mssv);
        printf("\t-Nhap ten: ");
        fflush(stdin);
        gets(x.ten);
        float tmp; //nhập cho trường kiểu số thực phải dùng biến tạm
        printf("\t-Nhap diem: ");
        scanf("%f",&tmp);
        x.diem=tmp;
}
void Xuat1SV(SV x)
{
        printf("%s\t%s\t%.2f",x.mssv,x.ten,x.diem);
}
void NhapSV(SV a[],int n)
{
        for (int i=0;i<n;i++)
        {
                printf("Nhap thong tin SV thu %d:\n",i+1);
                Nhap1SV(a[i]);
                printf("\n---------\n");
        }
```

```
}
void XuatSV(SV a[],int n)
{
	printf("MSSV\tTEN\tDIEM\n");
	for (int i=0;i<n;i++)
	{
		Xuat1SV(a[i]);
		printf("\n");
	}
}
```

- Bước 4: Viết hàm nhập xuất mảng (nếu đề yêu cầu thực hiện trên mảng):

```
void Nhapmang(int a[],int n)
{
	for (int i=0;i<n;i++)
	{
		printf("Nhap phan tu a[%d] = ",i+1);
		scanf("%d",&a[i]);
	}
}
void Xuatmang(int a[],int n)
{
	for (int i=0;i<n;i++)
		printf("%d\t",a[i]);
}
```

- Bước 5: Nhập số lượng (phần tử, sinh viên, hàng hoá,…):

```
void NhapN(int &n)
{
	do
	{
		printf("Nhap so luong N = ");
		scanf("%d",&n);
		if (n<1 || n>MAX_ARR) printf("Ban nhap sai. Moi nhap lai.\n");
	}
	while (n<1 || n>MAX_ARR); //nhớ phải kiểm tra điều kiện nhé
}
```

- Bước 6: Gõ thuật toán sắp xếp cần sử dụng.
- Bước 7: Chương trình chính.

```
int main() //bắt đầu chương trình chính
{
	clrscr(); //xoá màn hình
	int n; int a[MAX_ARR]; //nếu SV thì khai báo: SV a[MAX_ARR];
	NhapN(n); //nhập giá trị cho N
	Nhapmang(a,n); //nếu là SV thì: NhapSV(a,n);
	printf("Mang vua nhap:\n");
```

```
        Xuatmang(a,n); //nếu là SV thì: XuatSV(a,n);
        //thực thi thuật toán sắp xếp nào thì gọi tên thuật toán đó
        //lưu ý với QuickSort thì truyền vào 3 tham số: QuickSort(a,0,n-1)
        //là vị trí bắt đầu và vị trí kết thúc
        BubbleSort(a,n);
        //thực thi các thuật toán khác tương tự
        //xuất kết quả sau khi sắp xếp
        printf("\nKet qua sau khi sap xep:\n");
        Xuatmang(a,n); //nếu là SV thì: XuatSV(a,n);
        getch(); //dừng màn hình
        return 0; //mã thoát hàm main
}
```

➔ Lưu ý trường hợp sắp xếp giảm dần thì ta đổi dấu biểu thức điều kiện; duyệt từ cuối mảng (một số bài) thì ta đổi vị trí bắt đầu, kết thúc của vòng for hoặc while.
VD: for (int i=0;i<n;i++) là duyệt từ đầu mảng; for (int i=n-1;i>=0;i--) là duyệt từ cuối mảng.

HƯỚNG DẪN CHI TIẾT ĐẾN ĐÂY MÀ CÁC BẠN VẪN KHÔNG LÀM ĐƯỢC NỮA THÌ BẠN "XÁC ĐỊNH" ĐI NHÉ ☺