Bài 3. GIẢI THUẬT TÌM KIẾM VÀ SẮP XẾP

I. Mục tiêu

Sau khi thực hiện xong bài thực hành, sinh viên sẽ thực hiện được:

- Vận dụng được giải thuật tìm kiếm được trên mảng một chiều (áp dụng trên kiểu số và kiểu chuỗi)
- Vận dụng được giải thuật sắp xếp trên mảng một chiều.

II. Tóm tắt

1. Giải thuật tìm kiếm tuần tự (tuyến tính)

```
int  LinearSearch(int a[], int n, int x)
{
  int  i = 0;
  while ((i < n) && (a[i] != x))
          i++;
if (i == n)
          return -1; // Tìm hết mảng, không thấy
else
  return i;  // a[i] là phần tử có khóa x
}
```

2. Tìm kiếm nhị phân

```
int  BinarySearch(int a[], int n, int x)
{
          int  left = 0, right = n-1, mid;
          do {
                    mid = (left + right)/2; // chia nguyên
                    if (x == a[mid])
                              return mid; // tìm thấy
                    else if (x < a[mid])
                              right = mid – 1;
                    else
                              left = mid + 1;
          } while (left <= right);
          return -1;  // không tìm thấy
}
```

3. **Sắp xếp chọn – Selection Sort**

Ý tưởng	Cài đặt

SelectionSort(A[], n)
{
// Bước 1
	void SelectionSort (int a[], int n) {

Gán i = 1; (phần tử đầu tiên của mảng đã cho)
// Bước 2
Tìm phần tử A[min] nhỏ nhất trong dãy đang xét từ A[i] đến A[n]
// Bước 3
Hoán vị A[min] và A[i]
// Bước 4
Nếu i < n thì gán i = i + 1
Lặp lại bước 2
Ngược lại: n-1 phần tử đã được sắp đúng vị trí Dừng.
}
	int csmin;
	// chỉ số phần tử có giá trị nhỏ nhất
	for (int i = 0; i < n; i++)
	{
	csmin = i;
	for (int j = i+1; j < n; j++)
	if (a[j] < a[csmin])
	csmin = j;
	// ghi nhận vị trí ptử nhỏ nhất
	// Hoán vị
	int tam = a[csmin];
	a[csmin] = a[i];
	a[i] = tam;
	} //ket thuc for i
	}//ket thuc ham

III.Nội dung thực hành

1. Viết chương trình cho phép nhập vào một mảng các số nguyên (tối đa 30 phần tử) và thực hiện các yêu cầu sau:
   - Xuất mảng đã nhập ra màn hình.
   - Tìm kiếm phần tử x bất kỳ có trong mảng hay không? Nếu có cho biết vị trí xuất hiện của x.
   - Giả sử x được xuất hiện nhiều lần trong mảng, hãy xuất ra các vị trí xuất hiện của x.
     + Ví dụ: 2 5 7 2 4 9 2 8
     + X = 2: vị trí xuất hiện của 2 là: 0 3 6
   - Vận dụng giải thuật selection sort, hãy sắp xếp dãy số theo thứ tự tăng dần.
   - Hãy xây dựng hàm cho phép tìm kiếm phần tử x có trong mảng hay không?

   Dưới đây là đoạn mã lệnh cho phép nhập xuất một mảng. Sinh viên hãy xây dựng thêm các hàm cần thiết để thực thi các yêu cầu trên.

```
#include "conio.h"
#include "stdio.h"

const int KTM=30; //toi da 30 phan tu
//khai bao ham con
 void NhapMang(int a[], int &n);
 void XuatMang(int a[], int n);

int main()
{
    int mang[KTM], sophantu;
    NhapMang(mang,sophantu);
    printf("\nMang nhap vao la: ");
    XuatMang(mang,sophantu);
    return 1;
}
void NhapMang(int a[], int &n)
{
    do{
        printf("\nNhap so phan tu mang :");
        scanf("%d",&n);
    }while(n<=0 ||n>KTM);
    //nhap gia tri cho cac phan tu mang
    for(int i=0;i<n;i++)
    {
        printf("\nNhap phan tu thu %d : ",i);
        scanf("%d",&a[i]);
    }
}
void XuatMang(int a[], int n)
{
    printf("\n");
    for(int i=0;i<n;i++)
        printf("%3d",a[i]);
    printf("\n");
}
```

```
#include <stdio.h>
#include <conio.h>

void LinearSearch(int a[], int n, int x){
	printf("\nNhap phan tu x: ");
	scanf("%d", &x);
	for(int i=0; i<n; i++)
		if(a[i]==x)
			printf("\nPhan tu %d tai vi tri %d", x, i);
		else
			printf("\nKhong tim thay phan tu %d ", x);
}
int BinarySearch(int a[], int n, int x){
	int left=0, right=n-1, mid;
	do{
		mid=(left+right)/2; //chia nguyen
		if(x==a[mid])
			return mid; //tim thay
		else if(x<a[mid])
				right=mid-1;
			else
				left=mid+1;
	}while(left<=right);
		return -1; //khong tim thay
}
void SelectionSort(int a[], int n){
	int csmin, tam;
	for(int i=0; i<n; i++){
		csmin=i;
		for(int j=i+1; j<n; j++)
			if(a[j]>a[csmin])
				csmin=j;
				tam=a[csmin];
				a[csmin]=a[i];
				a[i]=tam;
	}
}
void Nhap(int a[], int &n){
	for(int i=0; i<n; i++){
		printf("\nNhap phan tu thu %d : ",i+1);
		scanf("%d", &a[i]);
```

```
	}
}
void Xuat(int a[], int n){
	for(int i=0; i<n; i++)
		printf("%3d", a[i]);
}
main(){
	int a[20], x, n;
	int chon;
	printf("Nhap so phan tu cua mang ");
	scanf("%d", &n);
	Nhap(a,n);
	printf("\nMang da nhap\n");
	Xuat(a,n);
	printf("\nBan chon chuc nang nao");
	printf("\n1.Sap xep mang giam dan\n");
	printf("\n2.Tim kiem tuyen tinh xuat vi tri cua x\n");
	printf("\n3.Tim kiem nhi phan\n");
	printf("\n4.Chon 4 thoat chuong trinh\n");
	scanf("%d", &chon);
	switch(chon){
		case 1:
			SelectionSort(a,n);
			printf("\nMang sau khi sap xep\n");
			Xuat(a,n);
		break;
		case 2:
			LinearSearch(a,n,x);
		break;
		case 3:
			printf("\nNhap phan tu x: ");
			scanf("%d", &x);
			if(BinarySearch(a, n, x)==-1)
				printf("\nKhong tim thay phan tu %d ", x);
			else
				printf("\nPhan tu %d tai vi tri thu %d ", x,BinarySearch(a, n,
x)+1);
			break;
	}
	getch();
}
```

2. Thông tin của một Ca học thực hành gồm có: mã lớp học phần, tên học phần, số sinh viên, phòng học. Yêu cầu:

- Xây dựng cấu trúc CATHUCHANH để biểu diễn thông tin cho một ca học thực hành.
- Nhập, xuất một danh sách các ca học thực hành.
- Tìm kiếm một ca học thực hành khi biết mã học phần.
- Tìm kiếm một ca học thực hành khi biết tên học phần.
- In ra danh sách các ca học thực hành khi biết phòng học.
- Sắp xếp danh sách ca học thực hành theo mã lớp học phần tăng dần.

IV.Bài tập làm thêm

Thông tin của một sản phẩm gồm có: mã sản phẩm (kiểu chuỗi 5 kí tự), tên sản phẩm (kiểu chuỗi 50 kí tự), đơn giá (số thực), ngày sản xuất (ngày, tháng, năm: kiểu số nguyên), số ngày sử dụng (kiểu số nguyên), nơi sản xuất (kiểu chuỗi 30 kí tự). Hãy viết chương trình thực hiện các yêu cầu sau:

- Nhập một danh sách các sản phẩm (tối đa 40 sản phẩm).
- Xuất danh sách đã nhập.
- Tìm kiếm sản phẩm khi biết mã sản phẩm.
- Tìm kiếm sản phẩm khi biết tên sản phẩm.
- In ra danh sách các sản phẩm khi biết nơi sản xuất.
- Sắp xếp danh sách sản phẩm theo thứ tự tăng dần của tên sản phẩm.