BÀI THỰC HÀNH

KỸ THUẬT TÌM KIẾM

BÀI TẬP ÔN TẬP

Viết các chương trình dạng hàm:

1. Nhập vào mảng một chiều số nguyên, in ra màn hình phần tử có giá trị nhỏ nhất *(khai báo mảng theo hai cách: kiểu tĩnh và kiểu động)*.
2. Nhập vào tọa độ 2 điểm trong mặt phẳng, tính khoảng cách giữa 2 điểm và in kết quả *(dùng kiểu dữ liệu có cấu trúc)*.
3. Nhập vào danh sách sinh viên, in ra những sinh viên có điểm trung bình >=5 và cho biết số lượng sinh viên nam. Thông tin của mỗi sinh viên gồm: mã số, họ tên, điểm trung bình và giới tính.

BÀI TẬP TÌM KIẾM TRÊN MẢNG

Bài 1: Viết chương trình cài đặt 2 giải thuật tìm kiếm: tuyến tính và nhị phân (*giả sử dãy số đầu vào có thứ tự tăng dần*).

Hướng dẫn: Xây dựng các hàm sau:

1. Tạo ngẫu nhiên mảng một chiều số nguyên có thứ tự tăng dần gồm N phần tử cho trước: **void PhatSinhMangTang(int a[], int N)**
2. Xem mảng phát sinh: **void XuatMang(int a[], int N)**
3. Tìm tuyến tính: **int TimTuyenTinh(int a[], int N, int X)**
4. Tìm nhị phân: **int TimNhiPhan(int a[], int N, int X)**
5. Hàm main():

- Phát sinh mảng tăng a với kích thước N cho trước (*không phải sắp xếp*).
- Xuất mảng xem kết quả phát sinh.
- Nhập giá trị cần tìm x.
- Tìm x theo 2 phương pháp.
- In kết quả tìm: Nếu tìm thấy thì cho biết vị trí tìm thấy, ngược lại in kết quả không tìm thấy cho từng phương pháp.

Bài 2: Viết chương trình cài đặt 2 giải thuật tìm kiếm: tuyến tính và nhị phân (*giả sử dãy số đầu vào có thứ tự tăng dần*). Yêu cầu chương trình phải xác định được số lần so sánh và vị trí tìm thấy (nếu có) của phần tử cần tìm.

Hướng dẫn:

1. Viết hàm tìm tuyến tính và tính số lần so sánh với phần tử cần tìm:
   int TimTuyenTinh(int a[], int N, int X, int &ss)
2. Tìm nhị phân và tính số lần so sánh với phần tử cần tìm:
   int TimNhiPhan(int a[], int N, int X, int &ss)
3. Hàm chính main():

- Phát sinh mảng tăng a với kích thước N cho trước (*không phải sắp xếp*).
- Xuất mảng xem kết quả phát sinh.
- Nhập giá trị cần tìm x
- Tìm x theo 2 phương pháp
- In kết quả tìm: Gồm vị trí (nếu tìm thấy x) và số lần so sánh cho từng phương pháp.

Bài 3: Cải tiến **Bài 2** sao cho: Nếu dãy không có thứ tự thì áp dụng phương pháp tìm tuyến tính, ngược lại dãy có thứ tự thì áp dụng phương pháp tìm nhị phân.

Hướng dẫn:

1. Viết hàm tìm nhị phân cho trường hợp dãy giảm dần (*trường hợp dãy tăng dần sử dụng lại hàm **TimNhiPhan** ở **Bài 3***):
   int TimNhiPhan2(int a[], int N, int X, int &ss)
2. Kiểm tra xem mảng có thứ tự tăng? (trả về ***true***: nếu tăng, ngược lại trả về ***false***)
   bool KiemTraTang(int a[], int N)
3. Kiểm tra xem mảng có thứ tự giảm? (trả về ***true***: nếu giảm, ngược lại trả về ***false***)
   bool KiemTraGiam(int a[], int N)
4. Phát sinh mảng ngẫu nhiên
   void PhatSinhMang(int a[], int N)
5. Hàm chính (main()):
   - Phát sinh mảng a với kích thước N cho trước.
   - Xuất mảng xem kết quả phát sinh.
   - Nhập giá trị cần tìm x
   - Kiểm tra nếu mảng có thứ tự tăng thì gọi hàm **TimNhiPhan**

Ngược lại, nếu mảng có thứ tự giảm thì gọi hàm **TimNhiPhan2**

Trường hợp còn lại thì gọi hàm **TimTuyenTinh** (*mảng không có thứ tự*)

- In kết quả như **Bài 2**

BÀI TẬP LÀM THÊM

Bài 1. Cài đặt thuật toán tìm tuyến tính cải tiến.

Bài 2. Cài đặt thuật toán tìm tuyến tính theo kỹ thuật đệ qui.

Bài 3. Cài đặt thuật toán tìm kiếm nhị phân theo kỹ thuật đệ qui.

BÀI THỰC HÀNH SỐ 2

KỸ THUẬT SẮP XẾP

(Số tiết: 3)

Biên soạn:

1. *Võ Quang Hoàng Khang*
2. *Nguyễn Thị Thanh Bình*

BÀI TẬP SẮP XẾP TRÊN MẢNG

Bài 1: Cài đặt các giải thuật sắp xếp cơ bản trên mảng các số nguyên, dữ liệu của chương trình được nhập vào từ file text được phát sinh ngẫu nhiên (số phần tử khoảng 10 000) và so sánh thời gian thực tế của các thuật toán:

1. **Chọn trực tiếp.**
2. **Chèn trực tiếp.**
3. **Đổi chỗ trực tiếp.**
4. **Nổi bọt.**
5. **Quicksort.**

BÀI TẬP LÀM THÊM

Bài 1. Cho mảng 1 chiều quản lý thông tin các sinh viên của 1 lớp học (tối đa 50 sinh viên). Mỗi sinh viên gồm các thông tin: MSSV, họ và tên, giới tính, địa chỉ và điểm trung bình. Viết chương trình thực hiện các yêu cầu sau:

1. Nhập các sinh viên vào danh sách.
2. In ra danh sách sinh viên.
3. Xóa 1 sinh viên với mã số x cho trước khỏi danh sách.
4. Sắp xếp danh sách sinh viên theo thứ tự tăng dần của điểm trung bình.
5. Sắp xếp danh sách sinh viên theo thứ tự tăng dần của họ và tên.

Hướng dẫn:

1. Khai báo cấu trúc thông tin sinh viên:

struct ttsinhvien

{ char MSSV[10], hoten[30];

int gioitinh; //1: nữ, 0: nam

char diachi[50];

float dtb;

};

typedef struct ttsinhvien SINHVIEN;

2. Viết các hàm sau:

void Nhap1SV(SINHVIEN &sv); //Nhập thông tin 1 sinh viên

void NhapDSSV(SINHVIEN dssv[], int &n); //Nhập danh sách sinh viên

void Xuat1SV(SINHVIEN sv); //Xuất thông tin 1 sinh viên

void XuatDSSV(SINHVIEN dssv[], int n); //Xuất danh sách sinh viên

int TimSV(SINHVIEN dssv[], int n, char maso[]); //Tìm sinh viên

void XoaSV(SINHVIEN dssv[], int n, char maso[]); //Hàm xóa

void SapTheoDTB(SINHVIEN dssv[], int n); //Sắp xếp theo điểm tb

void SapTheoHoTen(SINHVIEN dssv[], int n); //Sắp xếp theo họ tên

void Hoanvi(SINHVIEN &a, SINHVIEN &b); // Hoán vị 2 sinh viên

Lưu ý: Dùng hàm ***stricmp()*** để so sánh 2 chuỗi

3. Hàm chính (main()):

- Nhập danh sách sinh viên.
- Xuất danh sách.
- Nhập mã số sinh viên (x) cần xóa.
- Xóa x.
- Xem kết quả sau khi xóa.
- Sắp xếp theo điểm trung bình, xuất và xem kết quả.
- Sắp xếp theo họ tên, xuất và xem kết quả.

Bài 2. Cài đặt các giải thuật sắp xếp nâng cao trên mảng số nguyên và mảng cấu trúc.