TH 1: TÌM KIẾM

NỘI DUNG

1. Ôn lại cách viết một chương trình C dạng hàm
2. Nắm vững cấu trúc dữ liệu mảng 1 chiều
3. Nắm vững giải thuật tìm kiếm
    a. Tìm kiếm tuyến tính
    b. Tìm kiếm nhị phân

BÀI TẬP

Bài 1. Viết chương trình thực hiện:

- Nhập mảng gồm N số nguyên (N>0).
- Xuất mảng ra màn hình.
- Tìm phần tử có giá trị X trong mảng, nếu có cho biết vị trí xuất hiện của X trong mảng (làm theo 2 cách tìm kiếm tuyến tính và tìm kiếm nhị phân).

Bài 2. Viết chương trình quản lý thư viện, thông tin mỗi cuốn sách gồm: mã sách (int), tên sách (char[40]), giá (float).

- Nhập danh sách gồm N cuốn sách
- Xuất danh sách các cuốn sách ra màn hình
- Tìm cuốn sách có mã là X (Làm theo 2 cách: tìm tuyến tính và tìm nhị phân)
- Tìm cuốn sách có tên là X (Làm theo 2 cách: tìm tuyến tính và tìm nhị phân)
- Tìm cuốn sách có giá cao nhất (nếu có nhiều sách có giá cao nhất trùng nhau thì xuất hết ra màn hình).

HƯỚNG DẪN

Bài 1.

Nhắc lại, cấu trúc chung của một chương trình C:

```
//Khai báo thư viện
#include<stdio.h>
```

```
…
//Khai báo hằng nếu có
#define Tên_hằng  Giá_trị
//-------------------------------------------------------------
//Các hàm dùng trong chương trình
Kiểu_dữ_liệu_trả_về  Tên_hàm(khai báo các tham số đầu vào)
{
        //code...
}
…
//-------------------------------------------------------------
//Hàm chính
void main()
{
        clrscr();  //xóa màn hình
        //khai báo các biến cần dùng
        …
        //cài đặt chương trình theo menu chức năng
        //dùng một biến nguyên để lưu công việc mà người dùng chọn
        int chon;
        do{
                clrscr();
                //nhập chọn lựa của người dùng
                printf("1: Nhap mang\n");
                printf("2: Xuat mang\n");
                printf("3: Tim X theo Phuong phap tim tuyen tinh\n");
                printf("4: Tim X theo Phuong phap tim nhi phan\n");
                printf("0: Thoat\n");
                printf("Hay chon cong viec:"); scanf("%d", &chon);
                //thực hiện công việc cho lựa chọn tương ứng
                switch (chon){
                        case 1:  //Gọi hàm Nhập mảng
                                …
                        break;
                        case 2:  //Gọi hàm xuất mảng
                                …
                        break;
                        case 3:
                                //Nhập số nguyên X cần tìm
                                …
                                //Gọi hàm tìm tuyến tính
                                …
                        break;
                        case 4:
                                //Nhập số nguyên X cần tìm
                                …
                                //Gọi hàm tìm nhị phân
                                …
                        default:
                                chon=0;
                        break;
```

```
                }
                getch();
        }while (chon!=0);
}
```

– Dựa vào cấu trúc trên hãy cài đặt chương trình đầy dủ cho bài 1, trong đó:

- **Hàm sinh mảng:**
  + Bước 1: Nhập số phần tử của mảng sao cho N>0, nếu nhập sai thì bắt buộc phải nhập lại
  + Bước 2: Nhập giá trị cho từng phần tử trong mảng, nhập cho a[i] với i=0, 1, 2.., n-1.
- **Hàm xuất mảng**
- **Hàm tìm phần tử X (tìm tuyến tính)**
  + Nếu tìm thấy X trong mảng, hàm trả về vị trí tìm thấy

    Nếu không tìm thấy, hàm trả về -1
  + Mẫu hàm:

    int TimX_TuyenTinh(int a[], int n, int X)
- **Hàm tìm phần tử X (tìm nhị phân)**
  + Nếu tìm thấy X trong mảng, hàm trả về vị trí tìm thấy

    Nếu không tìm thấy, hàm trả về -1.
  + Mẫu hàm:

    int TimX_NhiPhan(int a[], int n, int X)

Lưu ý: tìm nhị phân chỉ áp dụng cho dãy đã được sắp tăng dần, nên khi tìm bằng phương pháp này, bạn phải nhập vào dãy số tăng dần hoặc sắp xếp lại dãy trước khi tìm.

Bài 2.

– Khai báo cấu trúc sách

```
typedef  struct  Tên_cấu_trúc
{
        //khai báo các biến thành phần của cấu trúc;
        …
}Tên_cấu_trúc_viết_gọn;


VD:
typedef struct CuonSach
{
        int  masach;
        char tensach[40];
        float gia;
}Sach;
```

- **Hàm nhập 1 cuốn sách:** nhập thông tin cho 1 cuốn sách
- **Hàm xuất 1 cuốn sách:** xuất thông tin của 1 cuốn sách
- **Hàm nhập danh sách các cuốn sách:** dùng 1 mảng một chiều để lưu danh các cuốn sách, mỗi phần tử trong mảng là 1 cuốn sách
    + Bước 1: Nhập số lượng cuốn sách (Nhập n)
    + Bước 2: Nhập thông tin cho từng cuốn sách (Nhập a[i], i=0, 1, ..n-1) bằng cách gọi hàm nhập 1 cuốn sách cho phần tử a[i]
- **Hàm xuất danh sách các cuốn sách:** Xuất thông tin từng cuốn sách a[i], i=0..n-1 bằng cách gọi hàm xuất 1 cuốn sách
- **Hàm tìm cuốn sách mã là X:** Làm theo hai cách tìm tuyến tính và nhị phân.
- **Hàm tìm cuốn sách có tên là X:** Làm theo hai cách tìm tuyến tính và nhị phân
    + Lưu ý: tên sách là kiểu chuỗi, do đó khi so sánh tên sách phải dùng hàm **strcmp().**
    + **VD:** Hàm strcmp(s1, s2): Hàm trả về 0 nếu chuỗi s1== chuỗi s2, trả về giá trị <0 nếu s1<s2 và >0 nếu s1>s2.

+ Hàm strcmp() nằm trong thư viện <string.h>

- **Hàm tìm cuốn sách giá lớn nhất:**

Bước 1: Tìm giá lớn nhất

Bước 2: Duyệt mảng sách, nếu cuốn sách nào có giá = giá lớn nhất (tìm được ở bước 1) thì xuất ra màn hình.

TH 2: SẮP XẾP

NỘI DUNG

Nắm vững các phương pháp sắp xếp:

1. Sắp xếp chọn (Selection sort)
2. Sắp xếp nổi bọt (Bubble sort)
3. Sắp xếp đổi chỗ trực tiếp (Interchange sort)
4. Sắp xếp chèn (Insertion sort)
5. Sắp xếp chèn dựa trên bước nhảy (Shell sort)
6. Sắp xếp nhanh (Quick sort)
7. Sắp xếp dựa trên cơ số (Radix sort)
8. Sắp xếp trộn (Merge sort)

BÀI TẬP

Bài 1. Viết chương trình thực hiện:

- Sinh mảng ngẫu nhiên gồm N số nguyên (N>0), mỗi phần tử có giá trị $\in(0, 100)$
- Xuất mảng ra màn hình
- Sắp xếp mảng tăng dần (giảm dần) bằng các thuật toán sắp xếp đã học

Yêu cầu: Viết chương trình theo menu chức năng (cho phép người dùng chọn lựa công việc cần thực hiện).

Bài 2. Viết chương trình quản lý nhân viên, thông tin mỗi nhân viên gồm: mã nhân viên, họ tên, lương.

- Nhập danh sách gồm N nhân viên
- Xuất danh sách nhân viên ra màn hình
- Sắp xếp danh sách tăng dần theo lương bằng thuật toán sắp xếp chọn
- Sắp xếp danh sách tăng dần theo họ tên bằng thuật toán sắp xếp chèn.

TH 3: DANH SÁCH - DANH SÁCH LIÊN KẾT ĐƠN

NỘI DUNG

1. Nắm vững cấu trúc danh sách

Danh sách là một kiểu dữ liệu trừu tượng gồm nhiều nút cùng kiểu dữ liệu, các nút trong danh sách có thứ tự.

Có hai cách cài đặt danh sách:

- Cài đặt theo kiểu kế tiếp, ta có danh sách kề: MẢNG MỘT CHIỀU
- Cài đặt theo kiểu liên kết, ta có các loại danh sách liên kết: DSLK ĐƠN, DSLK KÉP, DSLK VÒNG.

2. Nắm vững cấu trúc danh sách liên kết đơn

Các thao tác cơ bản trên dslk đơn:

1. Khởi tạo danh sách
2. Tạo một nút có dữ liệu X
3. Thêm một nút vào đầu danh sách
4. Thêm một nút vào sau một nút cho trước
5. Xóa nút đầu danh sách
6. Xóa nút đứng sau một nút cho trước
7. Xóa toàn bộ danh sách
8. Duyệt danh sách

BÀI TẬP

Bài 1. Viết chương trình tạo ra một danh sách có mỗi phần tử là một số nguyên. Chương trình có các chức năng sau:

1. Thêm phần tử mới vào cuối danh sách.
2. Thêm phần tử mới vào đầu danh sách.
3. Xuất danh sách ra màn hình

Bài 2. Làm tiếp bài 1 bổ sung các chức năng sau:

4. Chèn phần tử có giá trị x vào:
    a. Sau các phần tử có giá trị y, (với x, y nhập từ bàn phím).
    b. Trước các phần tử có giá trị y.
5. Cho biết giá trị của node thứ k trong danh sách (k bắt đầu từ 0).
6. Đếm số nút trên danh sách.
7. Tìm phần tử lớn nhất (nhỏ nhất) trong danh sách.
8. Xóa 1 phần tử có khóa là x.
9. Sắp xếp danh sách tăng dần theo phương pháp Interchange Sort.
10. *Cho biết các phần tử trong danh sách có được sắp xếp tăng dần/ giảm dần?
11. *Cho biết các phần tử dương trong danh sách có được sắp xếp tăng?
12. *Cho biết các phần tử chẵn và lẻ có xuất hiện xen kẽ không?
13. Cho danh sách (h,t). Hãy tạo ra 2 danh sách (h1,t1) chứa các giá trị dương và (h2,t2) chứa các giá trị âm trong danh sách đã cho.
14. * Sắp xếp các phần tử dương trong danh sách tăng dần.

HƯỚNG DẪN

Hướng dẫn sau đây chỉ có tính chất minh họa, bạn có thể làm theo cách khác.

Bài 1.

- Khai báo cấu trúc node:

```
struct node
{
        datatype info;      //lưu thông tin của mỗi phần tử trong danh sách
        struct node* next;  //lưu địa chỉ của nút tiếp theo
};
typedef struct node Node;
```

Vì danh sách có mỗi phần tử là một số nguyên nên kiểu dữ liệu của info là **int**.

Cần cài đặt các hàm sau:

1. Khởi tạo danh sách liên kết: *void init (Node* &phead)*
2. Kiểm tra danh sách có rỗng hay không: *int isEmpty(Node* phead)*
3. Tạo một node có dữ liệu X: *Node* createNode(int x)*
4. Thêm phần tử mới vào đầu danh sách: *void insertFirst(Node* &phead, int x)*
5. Thêm phần tử mới vào cuối danh sách: *void insertLast(Node* &phead, int x)*
6. Xuất danh sách ra màn hình: *void showList(Node* phead)*

- Hàm main() cài đặt theo menu chức năng.

```
//Hàm chính
void main()
{
	clrscr();
	//khai báo các biến quản lý danh sách
	Node* phead; //biến trỏ đến nút đầu tiên trong danh sách
	init(phead); //khởi tạo danh sách liên kết ban dầu chưa có nút nào

	//dùng một biến nguyên để lưu công việc mà người dùng chọn
	int chon;
	do{
		clrscr();
		//nhập chọn lựa của người dùng
		printf("1: Them 1 phan tu vao dau\n");
		printf("2: Them 1 phan tu vao cuoi\n");
		printf("3: Xuat danh sach\n");
		printf("0: Thoat\n");
		printf("Hay chon cong viec:"); scanf("%d", &chon);
		//thực hiện công việc cho lựa chọn tương ứng
		switch (chon){
			case 1:  /Thêm đầu
				int x;
				printf("Nhap gia tri phan tu can them:"); scanf("%d", &x);
			insertFirst(phead, x);
			break;
			case 2:  //Thêm cuối
				printf("Nhap gia tri phan tu can them:"); scanf("%d", &x);
			insertLast(phead, x);
			break;
```

```
                    case 3:             //Xuất danh sách
                            showList(phead);
                    break;
                    default: chon=0;
                    break;
                }
                getch();
        }while (chon!=0);
}
```

Bài 2: Bạn tự làm :).

TH 4: DANH SÁCH - DSLK ĐƠN

(TIẾP THEO)

NỘI DUNG

Cài đặt danh sách liên kết đơn, mỗi phần tử có kiểu struct.

BÀI TẬP

Viết chương trình quản lý danh sách sinh viên (sử dụng DSLKĐ), thông tin mỗi sv gồm: Mã sv - chuỗi tối đa 10 kí tự, Họ tên - chuỗi tối đa 40 kí tự, Điểm trung bình - số thực. Chương trình có các chức năng sau:

1) Nhập danh sách sinh viên.
2) Xuất danh sách sinh viên .
3) Xuất thông tin các sv có DTB>5.
4) Tìm sinh viên có mã là X.
5) Sắp xếp danh sách tăng dần theo điểm trung bình
6) Thêm một sinh viên vào sau sinh viên có tên là X. (Vận dụng hàm tìm SV có tên X vào hàm thêm một nút vào sau nút q)
7) Tìm SV có tên là X
8) Xóa SV đầu danh sách
9) Xóa SV cuối danh sách
10) Xóa toàn bộ danh sách
11) Xóa SV có tên là X
12) Xóa một sinh viên sau sinh viên có tên là X
13) Xóa tất cả các sinh viên có tên là X.
14) Sắp xếp DSSV tăng dần theo Mã sinh viên
15) In danh sách các SV được xếp loại khá.
16) Cho biết SV có điểm trung bình cao nhất / thấp nhất.
17) Cho biết SV có điểm trung bình thấp nhất trong số các SV xếp loại giỏi

HƯỚNG DẪN

Hướng dẫn sau đây chỉ có tính chất minh họa, bạn có thể làm theo cách khác.

- Khai báo cấu trúc node:

```
//ĐỊNH NGHĨA KIỂU DỮ LIỆU SINH VIÊN
struct SinhVien
{
        char masv[10];
        char hoten[40];
        float dtb;
} ;
typedef struct SinhVien SV;
struct node
{
        SV info;       //lưu thông tin của mỗi phần tử trong danh sách là 1 SV
        struct node* next;  //lưu địa chỉ của nút tiếp theo
};
typedef struct node Node;
```

- Để làm câu 1, 2 cần cài đặt các hàm sau (lưu ý rằng mỗi phần tử trong danh sách là một sinh viên nên kiểu dữ liệu của info là SV).
  1. Nhập thông tin cho 1 sinh viên: *nhap1SV(SV &x)*
  2. Xuất thông tin cho 1 sinh viên: *xuat1SV(SV x)*
  3. Khởi tạo danh sách liên kết: *void init (Node* &phead)*
  4. Kiểm tra danh sách có rỗng hay không: *int isEmpty(Node* phead)*
  5. Tạo một node có dữ liệu X: *Node* createNode(SV x)*
  6. Thêm phần tử mới vào đầu danh sách: *void insertFirst(Node* &phead, SV x)*

7. Thêm phần tử mới vào cuối danh sách: *void insertLast(Node* &phead, SV x)*
8. Xuất danh sách ra màn hình: *void showList(Node* phead).*

- Gọi thực hiện trong main()

Cách 1: Hàm main() đơn giản chỉ chạy thử các hàm con vừa code:
- Nhập danh sách sinh viên, giả sử số lượng sinh viên đã xác định (số lượng sinh viên do người dùng nhập từ bàn phím:

```
void main()
{
clrscr();
//khai báo các biến quản lý danh sách
Node* phead; //biến trỏ đến nút đầu tiên trong danh sách
init(phead); //khởi tạo danh sách liên kết ban dầu chưa có nút nào
SV x; //biến lưu thông tin của 1 sv dùng khi tạo 1 nút mới thêm vào ds

//Câu 1: Nhập 1 danh sách sinh viên
int n;
printf("Nhap so luong sinh vien:"); scanf("%d", &n);
for(int i=0; i<n; i++)
{
        SV x;
        printf("Nhap thong tin SV can them:"); nhap1SV(x);
        insertFirst(phead, x); // hoặc insertLast(phead, x); có gì khác nhau???
}
//Câu 2: Xuất danh sách sinh viên
showList(phead);
getch();
}
```

Cách 2: Hàm main() viết theo menu chức năng:
Khi nhập danh sách, số lượng sinh viên không cần biết trước, muốn tạo danh sách có bao nhiêu sinh viên chỉ cần chọn chức năng thêm bấy nhiêu lần.

```
//Hàm chính
void main()
{
        clrscr();
        //khai báo các biến quản lý danh sách
        Node* phead; //biến trỏ đến nút đầu tiên trong danh sách
        init(phead); //khởi tạo danh sách liên kết ban dầu chưa có nút nào
        SV x; //biến lưu thông tin của 1 sv dùng khi tạo 1 nút mới thêm vào ds
```

```
        //dùng một biến nguyên để lưu công việc mà người dùng chọn
        int chon;
        do{
                clrscr();
                //nhập chọn lựa của người dùng
                printf("1: Them 1 phan tu vao dau\n");
                printf("2: Them 1 phan tu vao cuoi\n");
                printf("3: Xuat danh sach\n");
                printf("0: Thoat\n");
                printf("Hay chon cong viec:"); scanf("%d", &chon);
                //thực hiện công việc cho lựa chọn tương ứng
                switch (chon){
                        case 1:  /Thêm đầu
                                printf("Nhap thong tin SV can them:"); nhap1SV(x);
                                insertFirst(phead, x);
                        break;
                        case 2:  //Thêm cuối
                                printf("Nhap thong tin SV can them:"); nhap1SV(x);
                                insertLast(phead, x);
                        break;
                        case 3:          //Xuất danh sách
                                showList(phead);
                        break;
                        default: chon=0;
                        break;
                }
                getch();
        }while (chon!=0);
}
```

Các câu sau bạn tự làm tương tự.

TH 5: DANH SÁCH - DSLK KÉP, VÒNG

BÀI TẬP

Bài 1. Cài đặt các thao tác trên DSLK kép, dữ liệu mỗi nút trong danh sách là một số nguyên.

Bài 2. Cài đặt các thao tác trên DSLK vòng, dữ liệu mỗi nút trong danh sách là một số nguyên.

TH 6: NGĂN XẾP (STACK)

NỘI DUNG

Stack có thể được xem là một dạng danh sách đặc biệt trong đó các tác vụ thêm vào hoặc xóa đi một phần tử chỉ diễn ra ở một đầu gọi là đỉnh *Stack*. Trên *Stack* các nút được thêm vào sau lại được lấy ra trước nên cấu trúc *Stack* hoạt động theo cơ chế vào sau ra trước - LIFO (Last In First Out).

Hai thao tác chính trên *Stack:*

- Tác vụ *push* dùng để thêm một phần tử vào đỉnh *Stack*
- Tác vụ *pop* dùng để xoá đi một phần tử ra khỏi đỉnh *Stack*.

BÀI TẬP

Câu 1: Hiện thực *Stac*k và các tác vụ của *Stack* bằng danh sách liên kết.

Câu 2: Viết chương trình đổi một số thập phân sang cơ số bất kỳ vận dụng *Stack*.

TH 7: HÀNG ĐỢI (QUEUE)

NỘI DUNG

Queue (hàng đợi) là một dạng danh sách đặt biệt, trong đó chúng ta chỉ được phép thêm các phần tử vào cuối hàng đợi và lấy ra các phần tử ở đầu hàng đợi. Vì phần tử thêm vào trước được lấy ra trước nên cấu trúc hàng đợi còn được gọi là cấu trúc FIFO(First In First Out). Hai tác vụ chính trên hàng đợi:

- insert – thêm nút mới vào cuối hàng đợi
- remove – dùng để xoá một phần tử ra khỏi hàng đợi.

BÀI TẬP

Viết chương trình quản lý việc khám bệnh cho một phòng khám tư. Biết rằng khi một bệnh nhân đến khám sẽ lấy một số thứ tự. Thông tin của mỗi bệnh nhân gồm: số thứ tự (chương trình tự sinh), họ tên, tuổi. Chương trình có các chức năng sau:

1. Thêm 1 bệnh nhân vào hàng đợi chờ khám, cho biết số thứ tự của người đó.
2. Xem bệnh nhân nào kế tiếp sẽ được khám
3. Xuất 1 bệnh nhân ra khỏi hàng đợi.
4. Xem tất cả danh sách bệnh nhân còn trong hàng đợi.
5. Xem còn bao nhiêu bệnh nhân chưa được khám?

HƯỚNG DẪN

1. Cài đặt cấu trúc dữ liệu Bệnh nhân: số thứ tự, họ tên, tuổi.
2. Cài đặt một Queue chứa các bệnh nhân chờ khám
3. Cài đặt các thao tác trên Queue
4. Cài đặt các chức năng theo mô tả của bài tập, chương trình phải viết theo menu chức năng.

TH 8: CÂY NHỊ PHÂN

BÀI TẬP

Cho trước 1 mảng a có n phần tử (mảng số nguyên/ mảng cấu trúc có một trường là khóa), hãy tạo một cây nhị phân có n node, mỗi nút lưu 1 phần tử của mảng.

1. Cài đặt hàm duyệt cây theo thứ tự: LNR, NLR, LRN, mức.
2. Tìm node có giá trị là X.
3. Xác định chiều cao của cây
4. Đếm số node trên cây.
5. Đếm số node lá
6. Đếm số node thỏa ĐK: đủ 2 cây con, có giá trị nhỏ hơn K, có giá trị lớn hơn giá trị của node con trái và nhỏ hơn giá trị của node con phải, có chiều cao cây con trái bằng chiều cao cây con phải.
7. Đếm số node lá có giá trị > X.
8. Cho biết node có giá trị lớn nhất/ nhỏ nhất.
9. Kiểm tra cây có cân bằng không?
10. Kiểm tra có là cây cân bằng hoàn toàn hay không?
11. Kiểm tra có phải là cây "đẹp" hay không? (mọi nút đều có đủ 2 nút con trừ nút lá).
12. Cho biết mọi node trên cây đều nhỏ hơn X (hoặc lớn hơn X) hay không?
13. Cho biết cây có phải là cây nhị phân tìm kiếm hay không?

TH 9: CÂY NHỊ PHÂN TÌM KIẾM, CÂN BẰNG (BST, AVL)

Cho trước 1 mảng a có n phần tử (mảng số nguyên/ mảng cấu trúc có một trường là khóa), hãy tạo một cây nhị phân có n node, mỗi nút lưu 1 phần tử của mảng.

1. Cài đặt hàm duyệt cây theo thứ tự: LNR, NLR, LRN, mức.
2. Tìm node có giá trị là X.
3. Xác định chiều cao của cây
4. Đếm số node trên cây.
5. Đếm số node lá
6. Đếm số node thỏa ĐK: đủ 2 cây con, có giá trị nhỏ hơn K, có giá trị lớn hơn giá trị của node con trái và nhỏ hơn giá trị của node con phải, có chiều cao cây con trái bằng chiều cao cây con phải.
7. Đếm số node lá có giá trị > X.
8. Cho biết node có giá trị lớn nhất/ nhỏ nhất.
9. Kiểm tra cây có cân bằng không?
10. Kiểm tra có là cây cân bằng hoàn toàn hay không?
11. Kiểm tra có phải là cây "đẹp" hay không? (mọi nút đều có đủ 2 nút con trừ nút lá).
12. Cho biết mọi node trên cây đều nhỏ hơn X (hoặc lớn hơn X) hay không?
13. Cho biết cây có phải là cây nhị phân tìm kiếm hay không?

TH 10: KIỂM TRA