Bài thực hành 1: TÌM KIẾM

1.1 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

1.2 BÀI TẬP

Bài 1. Viết chương trình thực hiện:

a) Nhập mảng gồm N số nguyên (N>0).
b) Xuất mảng ra màn hình.
c) 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).

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

1.3 HƯỚNG DẪN

Bài 1:

Chương trình có thể tổ chức gồm các hàm sau:

```
//Khai báo thư viện
#include<stdio.h>
#include<conio.h>
#include<stdlib.h> //thư viện chứa hàm random()
…
//------------------------------------------------------------
//Các hàm con dùng trong chương trình
void nhapn (int &n){...}
void SinhMang(int a[], int n){...}
void XuatMang(int a[], int n){...}
int TimTuyenTinh(int a[], int n, int X){...}
void SapXep(int a[], int n){...}
int TimNhiPhan(int a[], int n, int X){..}
//------------------------------------------------------------
//Hàm chính
void main()
{
        …
}
```

Hướng dẫn cài đặt cụ thể từng câu như sau.

Câu a. Bạn cài đặt 2 hàm sau:

1. Hàm nhập một số nguyên *n* (để nhập số lượng phần tử cho mảng), sử dụng cấu trúc *do..while* để bắt người dùng nhập 0 < *n* < 50.

```
void nhapn (int &n){
        do{
                //nhập n
                ...
                //nếu (n<=0 || n>=50) thì thông báo cho người dùng biết hãy nhập lại
                ...
        }while (n<=0 || n>=50);
}
```

2. Hàm nhập mảng: Nhập giá trị cho từng phần tử trong mảng, nhập cho a[i] với i = 0, 1, 2.., n-1.

Câu b. Viết hàm xuất mảng: void XuatMang(int a[], int n){...}

Sau khi cài đặt các hàm cho câu a và b, bạn hãy viết hàm chính main(), gọi thực hiện nhập mảng và xuất mảng. Chạy thử và sửa lỗi nếu có. Sau đó mới làm tiếp câu c.

Câu c. Tìm phần tử X trong mảng.

Nếu tìm thấy X trong mảng, trả về vị trí tìm thấy.

Nếu không tìm thấy, trả về -1.

Cách 1: Tìm kiếm tuyến tính

int TimTuyenTinh(int a[], int n, int X){ *//bạn tự code* }

Cách 2: Tìm kiếm nhị phân. Lưu ý tìm kiếm nhị phân chỉ áp dụng trên dãy đã được sắp xếp (tăng/giảm). Do đó, khi nhập dãy số để test, bạn hãy nhập dãy số tăng dần.

void SapXep(int a[], int n) { *//bạn tự code* }

int TimNhiPhan(int a[], int n, int X) { *//bạn tự code* }

Cách gọi thực hiện tìm kiếm trong hàm main():

```
void main()
{
...
int x;
printf("Nhap gia tri can tim: "); scanf("%d", &x);
//gọi thực hiện tìm kiếm x bằng phương pháp tìm nhị phân
int kq=TimTuyenTinh(a,n,x);
if (kq==-1) printf("Khong tim thay");
else printf("Tim thay tai vi tri %d", kq);
//gọi thực hiện tìm kiếm x bằng phương pháp tìm nhị phân tương tự
sapxep(a,n);
printf("Mang sau khi sap tang dan:"); XuatMang(a,n);
kq=TimNhiPhan(a,n,x);
//... tương tự ...
...
}
```

Mở rộng: Sau khi test thử hết các chức năng, chương trình chạy ra kết quả đúng, bạn hãy viết lại hàm main() theo menu chức năng.

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;
```

Chương trình có thể tổ chức gồm các hàm sau:

void nhap1Sach(Sach &x)

void xuat1Sach(Sach x)

void nhapn(int &n)

void nhapDS(Sach a[], int n)

void xuatDS(Sach a[], int n)

int timTuanTu(Sach a[], int n, int X)

int timNhiPhan(Sach a[], int n, int X)

- **Hàm nhập 1 cuốn sách:** nhập thông tin cho 1 cuốn sách

  void nhap1Sach(Sach &x)

- **Hàm xuất 1 cuốn sách:** xuất thông tin của 1 cuốn sách

  void xuat1Sach(Sach x)

- **Hàm nhập số lượng phần tử của danh sách:** nhập số lượng cuốn sách

  void nhapn(int &n)

- **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.

```
void nhapDS(Sach a[], int n)
{
        //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]
        for(int i=0; i<n; i++)
                nhap1Sach(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.

  int timTuanTu(Sach a[], int n, int X)

  int timNhiPhan(Sach a[], int n, int X)

- **Hàm xuất ra các cuốn sách có tên là Y:** Duyệt danh sách, nếu gặp cuốn nào có tên là Y thì xuất thông tin cuốn sách đó ra màn hình.
- **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.

1.4 NÂNG CAO

Cải tiến giải thuật tìm kiếm tuyến tính bằng kỹ thuật đặt lính canh.

Hãy làm lại bài 2 với yêu cầu mã sách có kiểu là chuỗi tối đa 10 ký tự.

Bài thực hành 2: SẮP XẾP

2.1 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)

2.2 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 lương bằng thuật toán sắp xếp chèn.

2.3 HƯỚNG DẪN

Bài 1. Bạn tự cài đặt các hàm:

+ Sinh mảng.
+ Xuất mảng.
+ Sắp xếp mảng bằng các giải thuật đã học.
+ Sau khi test hết các hàm đã chạy ra kết quả đúng, tổ chức lại hàm main() theo menu chức năng như sau:

```
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: Sinh mang\n");
                printf("2: Xuat mang\n");
                printf("3: Interchange Sort\n");
                printf("4: Bubble Sort\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 Sinh mảng
                                …
                        break;
                        case 2:         //Gọi hàm xuất mảng
                                …
                        break;
                        case 3:
                                //Gọi hàm sắp xếp bằng Interchange
                                …
                                //Xuất mảng để xem kết quả sau khi sắp
                                printf("Mang sau khi sap la: \n");
                                XuatMang(a,n);
                        break;
                        case 4:
                                //Gọi hàm sắp xếp bằng Bubble
                                …
                                //Xuất mảng để xem kết quả sau khi sắp
                                printf("Mang sau khi sap la: \n");
```

```
                        XuatMang(a,n);
                    break;
                    case 5: ... //tương tự
                        …
                    default:
                        chon=0;
                    break;
                }
                getch();
            }while (chon!=0);
}
```

Bài 2: Bạn tự code tương tự bài 2 trong bài thực hành số 1.

2.4 NÂNG CAO

Làm lại bài 2, trong đó cài đặt chương trình theo menu chức năng. Trong đó, cài đặt sắp xếp danh sách giảm dần *theo họ tên* bằng tất cả các phương pháp sắp xếp đã học (mỗi phương pháp tương ứng một chức năng trong menu).

Bài thực hành 3: DANH SÁCH - DANH SÁCH LIÊN KẾT

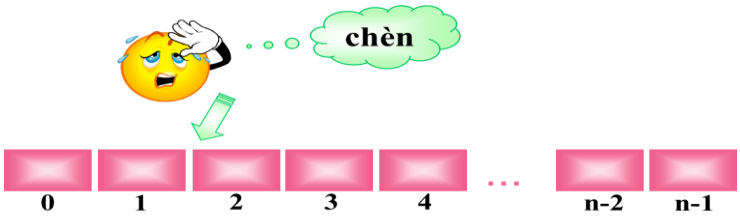
3.1 NỘI DUNG

1. Nắm vững cấu trúc danh sách

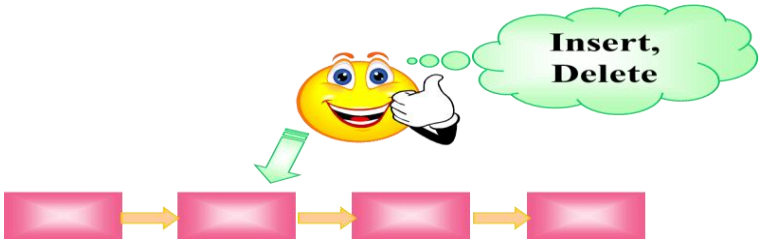
Danh sách là một kiểu dữ liệu trừu tượng gồm nhiều nút cùng kiểu dữ liệu, các nút trong danh sách có thứ tự.

Có hai cách cài đặt danh sách:

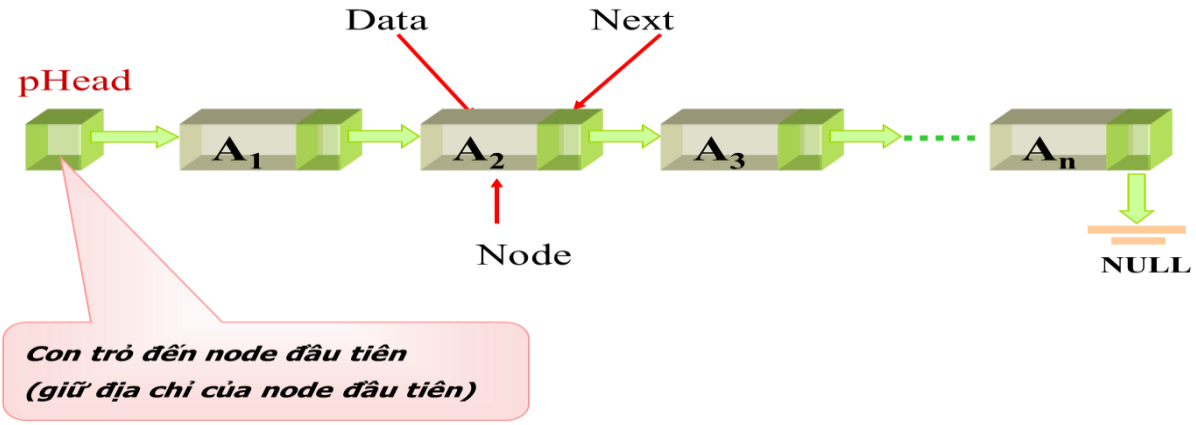
- Cài đặt theo kiểu kế tiếp, ta có danh sách kề: MẢNG MỘT CHIỀU



- 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

3.2 BÀI TẬP

Bài 1. Sử dụng cấu trúc dữ liệu danh sách liên kết đơn để lưu một dãy số nguyên, viết chương trình thực hiện:

1) Nhập vào một dãy số nguyên
2) Xuất dãy số nguyên ra màn hình

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

3) Chèn phần tử có giá trị y vào:
    a. Sau các phần tử có giá trị x, (với x, y nhập từ bàn phím).
    b. Trước các phần tử có giá trị x.

4) Đếm số nút trên danh sách.
5) Cho biết giá trị của node thứ k trong danh sách (k bắt đầu từ 0).
6) Tìm phần tử lớn nhất (nhỏ nhất) trong danh sách.
7) Xóa 1 phần tử có khóa là x.
8) Sắp xếp danh sách tăng dần theo phương pháp Interchange Sort.

3.3 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;  //biến con trỏ quản lý, 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)*

Cách 1 Nhập danh sách với số lượng phần tử biết trước, số lượng phần tử do người dùng

nhập.

```
void input(Node* &phead)
{
        init(phead); //khởi tạo danh sách liên kết ban dầu chưa có nút nào
        //tạo danh sách gồm n số nguyên
      int n, x;
      printf("Nhap so luong phan tu: "); scanf("%d", &n);
      for(int i=0; i<n; i++)
      {
              printf("Nhap so can them vao danh sach: ");
              scanf("%d", &x);
              insertFirst(phead, x);
      }
}
```

- Bạn hãy chạy thử với n = 3, thêm lần lượt x = 1, x = 2, và x = 3. Quan sát kết quả và cho nhận xét.
- Thay gọi hàm ***insertFirst(phead, x)*** trong vòng for bằng hàm ***insertLast(phead, x)***, chạy thử với n=3, thêm lần lượt x= 1, x=2, và x=3. Quan sát kết quả và cho nhận xét.

Cách 2: Nhập vào một dãy số nguyên đến khi gặp số 0 thì dừng.

```
void input(Node* &phead)
{
	int x; init (phead);
	do{
		printf("Nhap gia tri x: "); scanf("%d", &x);
		if(x!=0)
		{
			Node* p=CreateNode(x);
			if (p!=NULL)
				InsertFirst(phead, x);
		}
	}while(x!=0);
}
```

Cách 3:

Hàm main() cài đặt theo menu chức năng cho phép nhập danh sách tùy ý người dùng, số lượng phần tử không biết trước. Khi nhập danh sách, muốn tạo danh sách có bao nhiêu phần tử 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
        //dùng một biến nguyên để lưu công việc mà người dùng chọn
        int chon, x;
        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 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.

Chèn phần tử có giá trị y vào sau các phần tử có giá trị x, (với x, y nhập từ bàn phím). Vận dụng 2 hàm:

- Tìm phần tử có giá trị x
- Thêm một phần tử vào sau nút p cho trước

Bạn tự làm tiếp!!!

3.4 NÂNG CAO

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

9) Xóa 1 phần tử có khóa là x.

10) Sắp xếp danh sách tăng dần theo phương pháp Interchange Sort.

11) Sắp xếp danh sách tăng dần theo phương pháp Selection Sort.

12) Cho biết các phần tử trong danh sách có được sắp xếp tăng dần/ giảm dần?

13) Cho biết các phần tử chẵn và lẻ có xuất hiện xen kẽ không?

Bài thực hành 4: DANH SÁCH - DSLK ĐƠN (TIẾP THEO)

4.1 NỘI DUNG

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

4.2 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 mã là X. (Vận dụng hàm tìm SV có mã X và hàm thêm một nút vào sau nút *p* cho trước).
7) Xóa SV đầu danh sách
8) Xóa SV cuối danh sách
9) Xóa toàn bộ danh sách
10) Xóa SV có mã là X

4.3 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.

4.4 NÂNG CAO

- Cài đặt tiếp các chức năng sau:
  11) Xóa một sinh viên sau sinh viên có mã là X
  12) Xóa tất cả các sinh viên có tên là X.
  13) Sắp xếp DSSV tăng dần theo Mã sinh viên
  14) In danh sách các SV được xếp loại khá.
  15) Cho biết SV có điểm trung bình cao nhất / thấp nhất.
  16) Cho biết SV có điểm trung bình thấp nhất trong số các SV xếp loại giỏi.
- Cài đặt các thao tác trên DSLK kép, dữ liệu mỗi nút trong danh sách là một số nguyên.
- Cài đặt các thao tác trên DSLK vòng, dữ liệu mỗi nút trong danh sách là một số nguyên.

Bài thực hành 5: NGĂN XẾP (STACK) & HÀNG ĐỢI (QUEUE)

5.1 NỘI DUNG

1. *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*.

2. *Queue* (hàng đợi) là một dạng danh sách đặc 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.

5.2 BÀI TẬP

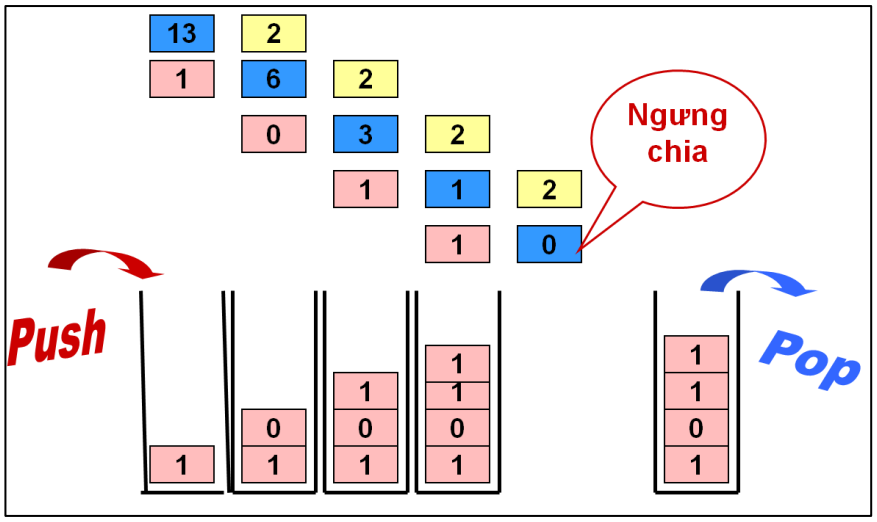
Bài 1: Viết chương trình đổi một số thập phân sang cơ số nhị phân vận dụng *Stack*.

Bài 2: 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) Cho biết thông tin bệnh nhân tiếp theo sẽ được khám là ai và xuất bệnh nhân đó ra khỏi hàng đợi.
3) Cho biết có bao nhiêu bệnh nhân chưa được khám?
4) Cho biết có bao nhiêu bệnh nhân đã được khám?

5.3 HƯỚNG DẪN

Bài 1: Sử dụng Stack để chứa số dư khi chia số thập phân cho 2. Như vậy, Stack sẽ chứa các bit của số ở dạng nhị phân.



Bước 1. Hiện thực *Stack* và các tác vụ của *Stack* bằng danh sách liên kết, mỗi phần tử của

Stack kiểu int.

1. Khai báo cấu trúc dữ liệu Stack. Thay đổi Datatype cho phù hợp yêu cầu của đề bài.

```
typedef struct node
{
        DataType      info;
        struct node * next;
}Node;
typedef  Node* STACK;
```

2. Cài đặt các hàm:

- Khởi tạo Stack: void Init(STACK &s)
- Kiểm tra Stack rỗng: int IsEmpty(STACK s)
- Tác vụ thêm một phần tử vào Stack: void Push(STACK &s, DataType x)
- Tác vụ lấy một phần tử ra khỏi Stack: int Pop(STACK &s, DataType &x)

Bước 2. Vận dụng Stack để đổi một số thập phân *n* sang nhị phân. Mã giả hàm đổi số như sau:

```
void Convert(int n, Stack &s){
        int sodu; //biến chứa số dư khi chia n cho 2
        Lặp chừng nào (n≠ 0)
        {       sodu=n%2;
                Bỏ sodu vào Stack;
                n=n/2;
        }
```

– Xuất số ở dạng nhị phân ra màn hình: Lấy lần lượt các bit trong Stack và xuất ra màn

hình.

– Hàm chính main()

```
//Hàm chính
void main()
{
	clrscr();
	//khai báo các biến quản lý Stack
	STACK s;
	init(s); //khởi tạo Stack ban đầu chưa có gì
	//Nhập số nguyên n cần đổi
	printf("Nhap so o he thap phan: "); scanf("%d", &n);
	//Gọi thực hiện việc đổi số
	Convert(n, s);
	//Xuất số ở dạng nhị phân ra màn hình
	...
	getch();
}
```

Bài 2:

– Cài đặt cấu trúc dữ liệu Bệnh nhân: số thứ tự, họ tên, tuổi. Họ tên và tuổi do người dùng nhập, số thứ tự do chương trình tự phát sinh theo số người đến khám.

– Cài đặt một Queue chứa các bệnh nhân chờ khám

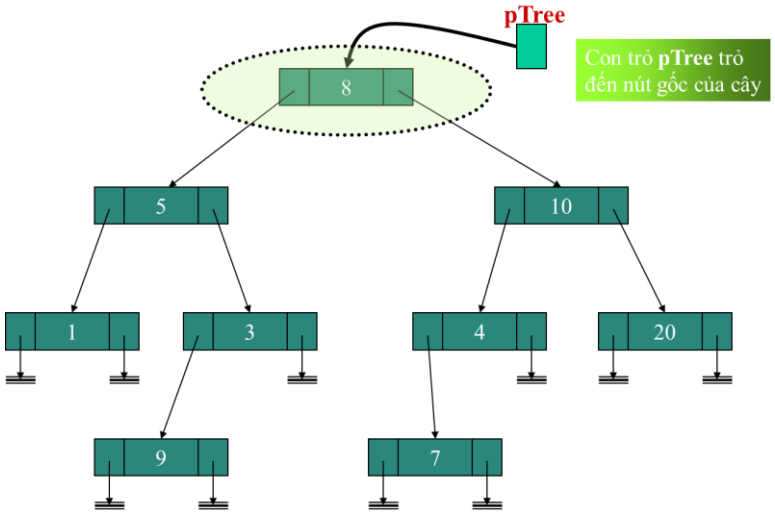
– Cài đặt các thao tác trên Queue

– Cài đặt các chức năng theo mô tả của bài tập, chương trình phải viết theo menu chức năng.

Bài thực hành 6: CÂY NHỊ PHÂN – CÂY NHỊ PHÂN TÌM KIẾM

6.1 NỘI DUNG

1. Nắm vững cấu trúc cây nhị phân



2. Cài đặt được một số thao tác cơ bản trên cây

6.2 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.

6.3 NÂNG CAO

Cài đặt cấu trúc dữ liệu liên kết cho cây nhị phân tìm kiếm, với các thao tác:

a) Cài đặt các thao tác xây dựng cây: Init, IsEmpty, CreateNode

b) Cài đặt thao tác cập nhật: Insert, Remove

ÔN TẬP –KIỂM TRA

Chúc các bạn thi tốt!!!