NỘI DUNG CHÍNH BUỔI 10

MẢNG 1 CHIỀU (tt)

(6) Các kỹ thuật xử lý trên mảng 1 chiều

1) Liệt kê giá trị các phần tử thỏa điều kiện:

Giải thuật:

- Khai báo và gán giá trị cho biến đếm_dk = 0
- Duyệt từ đầu => cuối mảng
  - Nếu phần tử hiện hành thỏa điều kiện
    - In giá trị phần tử hiện hành
    - đếm_dk ++
- Nếu đếm_dk = 0
  - In thông báo không tồn tại phần tử thỏa điều kiện trong mảng

VD: Liệt kê giá trị các phần tử chẵn ∈ [20, 40] trong mảng

```
void LietKeChanTu20Den40(int a[],int n)
{
   int dem_chan_20_40=0;
   for(int i=0;i<n;i++)
        if(a[i]%2==0&&a[i]>=20&&a[i]<=40)
        {
             printf("%d\t",a[i]);
             dem_chan_20_40++;
        }
   if(dem_chan_20_40==0)
        printf("Khong co phan tu chan tu 20 den 40 trong mang");
}
```

2) Đếm số lượng các phần tử thỏa điều kiện:

Giải thuật:

- Khai báo và gán giá trị cho biến đếm_dk = 0
- Duyệt từ đầu => cuối mảng
  - Nếu phần tử hiện hành thỏa điều kiện
    - đếm_dk ++
- Nếu đếm_dk > 0
  - In thông báo có đếm_dk phần tử thỏa điều kiện trong mảng

  Ngược lại,
  - In thông báo không tồn tại phần tử thỏa điều kiện trong mảng

VD: Đếm số lượng các phần tử lẻ có ký số cuối là 3 hoặc 5 trong mảng

```
void DemLeKySoCuoi3Hoac5(int a[],int n)
{
   int dem_le_ky_so_cuoi_3_5=0;
   for(int i=0;i<n;i++)
        if(a[i]%2!=0&&(a[i]%10==3||a[i]%10==5))
             dem_le_ky_so_cuoi_3_5++;
   if(dem_le_ky_so_cuoi_3_5>0)
        printf("Co %d phan tu le co ky so cuoi la 3 hoac 5 trong
                mang", dem_le_ky_so_cuoi_3_5);
   else
        printf("Khong co phan tu le co ky so cuoi la 3 hoac 5
               trong mang");
}
```

3) Tính tổng giá trị các phần tử thỏa điều kiện:

Giải thuật:

- **Khai báo và gán giá trị cho biến tổng_đk = 0**
- **Duyệt từ đầu => cuối mảng**
  - **Nếu phần tử hiện hành thỏa điều kiện**
    - **tổng_đk+=a[i]**
- **Nếu tổng_đk > 0**
  - **In thông báo tổng giá trị các phần tử thỏa điều kiện trong mảng là tổng_đk**

Ngược lại,

- **In thông báo không tồn tại phần tử thỏa điều kiện trong mảng**

VD: Tính tổng giá trị các phần tử là số nguyên tố > 50 trong mảng

```
void TongSNTLonHon50(int a[],int n)
{
   int tong_snt_lon_hon_50=0;
   for(int i=0;i<n;i++)
        if(KiemTraSNT(a[i])==1&&a[i]>50)
             tong_snt_lon_hon_50+=a[i];
   if(tong_snt_lon_hon_50>0)
        printf("Tong gia tri cac phan tu la SNT lon hon 50 trong
                mang la %d",tong_snt_lon_hon_50);
   else
        printf("Khong co phan tu la SNT lon hon 50 trong mang");
}
```

4) Tính trung bình cộng giá trị các phần tử thỏa điều kiện:

Giải thuật:

- Khai báo và gán giá trị cho biến tổng_đk = 0, đếm_đk = 0
- Duyệt từ đầu => cuối mảng
  - Nếu phần tử hiện hành thỏa điều kiện
    - tổng_đk+=a[i]
    - đếm_đk ++
- Nếu tổng_đk > 0 hoặc đếm_đk > 0
  - In thông báo trung bình cộng giá trị các phần tử thỏa điều kiện trong mảng là tổng_đk / đếm_đk (ép sang kiểu số thực)

  Ngược lại,
  - In thông báo không tồn tại phần tử thỏa điều kiện trong mảng

VD: Tính trung bình cộng giá trị các phần tử là số chính phương chẵn trong mảng

```
void TBCongSCPChan(int a[],int n)
{
   int tong_scp_chan=0,dem_scp_chan=0;
   for(int i=0;i<n;i++)
        if(KiemTraSCP(a[i])==1&&a[i]%2==0)
        {
             tong_scp_chan+=a[i];
             dem_scp_chan++;
        }
   if(tong_scp_chan>0)
        printf("Trung binh cong gia tri cac phan tu la SCP chan
            trong mang la %d",(float)tong_scp_chan/dem_scp_chan);
   else
        printf("Khong co phan tu la SCP chan trong mang");
}
```

5) Kiểm tra mảng thỏa điều kiện hay không:

Giải thuật "đặt cờ hiệu":

- **Khai báo biến cờ hiệu**
- **Xét các trường hợp làm thay đổi giá trị biến cờ hiệu**
- **Trả về giá trị biến cờ hiệu**

VD1: Viết hàm kiểm tra xem mảng các số nguyên có thứ tự tăng dần không? (Trả về 1 nếu mảng tăng dần, ngược lại trả về 0).

```
int KiemTraTang(int a[],int n)
{
   int mangtang=1;
   for (int i=0;i<n-1;i++)
        if (a[i]>a[i+1]) //Vi phạm điều kiện tăng dần
        {
             mangtang=0;
             break;
        }
   return mangtang;
}
```

VD2: Viết hàm kiểm tra xem trong mảng các số nguyên có tồn tại số nguyên lẻ lớn hơn 100 hay không? (Trả về 1 nếu có tồn tại số lẻ và lớn hơn 100, ngược lại trả về 0).

```
int KiemTraLe(int a[],int n)
{
   int cophantulelonhon100=0;
   for(int i=0;i<n;i++)
        if(a[i]%2!=0&&a[i]>100) //Gặp phần tử thoả
        {
             cophantulelonhon100=1;
             break;
        }
   return cophantulelonhon100;
}
```

6) Tìm kiếm phần tử trong mảng thỏa điều kiện:

Giải thuật "đặt lính canh":

- **Chọn phần tử đầu mảng làm lính canh**
- **Lần lượt duyệt qua các phần tử thứ 2 trở đi => cuối mảng**
  - **Nếu phần tử đang xét thỏa điều kiện tìm kiếm hơn**
    - **Chọn phần tử đang xét làm lính canh mới**
- **Trả về vị trí / giá trị của phần tử làm lính canh mới nhất sau cùng.**

VD3: Viết hàm tìm vị trí và giá trị phần tử lớn nhất trong mảng một chiều các số nguyên.

```
void TimMax(int a[],int n)
{
   int max=a[0],vtmax=0;
   for(int i=1;i<n;i++)
        if(a[i]>max)
        {
             max=a[i];
             vtmax=i;
        }
   printf("Phan tu lon nhat trong mang la a[%d] = %d",vtmax,max);
}
```

VD4: Viết hàm tìm vị trí phần tử lớn nhất trong mảng một chiều các số nguyên.

```
void TimVTMax(int a[],int n)
{
   int max=a[0],vtmax=0;
   for(int i=1;i<n;i++)
        if(a[i]>max)
        {
             max=a[i];
             vtmax=i;
        }
   return vtmax;
}
```

VD5: Viết hàm tìm giá trị phần tử lớn nhất trong mảng một chiều các số nguyên.

```
void TimGTMax(int a[],int n)
{
   int max=a[0];
   for(int i=1;i<n;i++)
        if(a[i]>max)
             max=a[i];
   return max;
 }
```

7) Sắp xếp mảng tăng / giảm dần:

- **Khái niệm nghịch thế:**

  Xét mảng các số a[0], a[1], … a[n-1].

  Nếu có **i<j** và **a[i] > a[j]**, thì ta gọi đó là một **nghịch thế**.

- Mảng chưa sắp xếp sẽ có nghịch thế => Tìm tất cả nghịch thế và triệt tiêu chúng bằng cách hoán vị (đổi chỗ) 2 phần tử tương ứng trong nghịch thế.
- Mảng đã có thứ tự sẽ không chứa nghịch thế.
- Có nhiều phương pháp sắp xếp được thực hiện bằng cách so sánh và đổi chỗ. Ở đây, ta sử dụng phương pháp sắp xếp đơn giản như sau: bắt đầu từ phần tử đầu tiên so sánh với các phần tử còn lại, nếu từng cặp phần tử được so sánh đứng sai vị trí sắp xếp (tạo nghịch thế) thì ta đổi chỗ 2 phần tử đó với nhau. Tiếp tục đến phần tử kế tiếp so sánh với các phần tử còn lại...cho đến khi gặp phần tử cuối cùng => quá trình sắp xếp mảng hoàn tất: mảng đã được sắp xếp tăng / giảm dần.

Giải thuật sắp xếp chọn trực tiếp Interchange Sort:

- **Duyệt lần lượt các phần tử thứ i từ đầu => kế cuối mảng**
  - **Với mỗi phần tử i hiện hành, duyệt lần lượt các phần tử thứ j từ sau i => cuối mảng**
    - **Nếu a[i] và a[j] tạo nghịch thế**
      - **Đổi chỗ a[i], a[j];**

VD6: Viết hàm sắp xếp mảng một chiều các số nguyên tăng dần.

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

8) Xóa phần tử tại vị trí vt cho trước trong mảng:

- Xóa phần tử tại vị trí vt (0 ≤ vt ≤ n-1, vt = 0: xóa phần tử đầu mảng, vt = n-1: xóa phần tử cuối mảng), khi đó các phần tử từ vị trí vt+1 đến n-1 bị dời lên phía trước 1 vị trí.

Giải thuật:

- **Duyệt qua các phần tử thứ vt trở đi => cuối mảng**
  - **Lần lượt thay nội dung của phần tử hiện hành bằng nội dung phần tử đứng kế sau nó**
- **Giảm số lượng phần tử thực có (n--).**

VD7: Viết hàm xóa phần tử đầu tiên trong mảng.

```
void XoaDau(int a[],int &n)
{
   for(int i=0;i<n-1;i++)
        a[i]=a[i+1];
   n--;
}
```

VD8: Viết hàm xóa phần tử tại vị trí vt cho trước trong mảng.

```
void XoaTaiViTri(int a[],int &n,int vt)
{
   for(int i=vt;i<n-1;i++)
        a[i]=a[i+1];
   n--;
}
```

9) Chèn thêm 1 phần tử có giá trị gt vào mảng tại vị trí vt cho trước:

- Thêm phần tử có giá trị gt tại vị trí vt ($0 \le vt \le n$, vt = 0: thêm phần tử vào đầu mảng, vt = n: thêm phần tử vào cuối mảng), khi đó các phần tử từ vị trí n-1 xuống vt bị dời xuống phía sau 1 vị trí để phần tử có giá trị gt được chèn thêm vào mảng tại vị trí vt.

Giải thuật:

- **Duyệt qua các phần tử thứ vt trở đi => cuối mảng**
  - **Lần lượt thay nội dung của phần tử hiện hành bằng nội dung phần tử đứng kế sau nó**
- **Giảm số lượng phần tử thực có.**

VD9: Viết hàm thêm phần tử có giá trị gt vào cuối mảng.

```
void ThemCuoi(int a[],int &n,int gt)
{
   a[n]=gt;
   n++;
}
```

VD10: Viết hàm thêm phần tử có giá trị gt vào mảng tại vị trí vt cho trước.

```
void ChenTaiViTri(int a[],int &n,int gt,int vt)
{
   for(int i=n-1;i>=vt;i--)
        a[i+1]=a[i] ;
   a[vt]=gt;
   n++;
}
```

```
//Hoặc viết cách khác:

void ChenTaiViTri(int a[],int &n,int gt,int vt)
{
   for(int i=n;i>vt;i--)
        a[i]=a[i-1] ;
   a[vt]=gt;
   n++;
}
```

10) Tìm 1 phần tử có giá trị gt trong mảng (gt là khóa tìm kiếm):

- **Tìm kiếm tuyến tính:** lần lượt so sánh giá trị phần tử cần tìm gt với từng giá trị của các phần tử thứ 0, 1, 2, … n-1 của mảng, cho đến khi gặp phần tử có khóa cần tìm (có giá trị = giá trị gt), hoặc đã tìm hết mảng mà không thấy phần tử có khóa cần tìm (có giá trị ≠ giá trị gt)

Giải thuật đặt cờ hiệu:

- **Khai báo và gán giá trị cho biến cờ hiệu timthay = -1 (giả sử chưa tìm thấy)**
  - **Duyệt từ đầu => cuối mảng**
    - **Nếu phần tử hiện hành có giá trị = gt**
      - **Gán giá trị biến cờ hiệu = vị trí phần tử hiện hành**
      - **Dừng quá trình duyệt**
- **Trả về giá trị biến cờ hiệu**

VD11: Viết hàm tìm 1 phần tử có giá trị gt sử dụng phương pháp tìm kiếm tuyến tính.

```
int TimKiemTuyenTinh(int a[], int n, int gt)
{
   int timthay=-1; //chưa tìm thấy
   for(int i=0;i<n;i++)
        if(a[i]==gt)
        {
             timthay=i; //tìm thấy tại vị trí i
             break;
        }
   return timthay;
}
```

- **Tìm kiếm nhị phân: chỉ áp dụng cho mảng đã có thứ tự tăng**. Ý tưởng của phương pháp là tại mỗi bước ta tiến hành so sánh gt với phần tử nằm ở vị trí giữa của dãy tìm kiếm hiện hành, dựa vào kết quả so sánh này để quyết định giới hạn dãy tìm kiếm kế tiếp là nửa trên hay nửa dưới của dãy tìm kiếm hiện hành.

Giải thuật đặt cờ hiệu:

- **Khai báo và gán giá trị cho biến dau (đầu), cuoi (cuối), giua (giữa)**
- **Khai báo và gán giá trị cho biến cờ hiệu timthay = -1 (giả sử chưa tìm thấy)**
  - **Trong khi dau ≤ giua**
    - **giua = (dau + cuoi) / 2**
    - **Nếu a[giua]=gt (giá trị cần tìm)**
      - **Gán giá trị biến cờ hiệu timthay = giua (tìm thấy tại vị trí giữa)**
      - **Dừng quá trình phân hoạch tìm kiếm**
    - **Nếu gt < a[giua]**
      - **cuoi = giua – 1 (chỉ tìm gt trong nửa trên của dãy)**
    - **Ngược lại,**
      - **dau = giua + 1 (chỉ tìm gt trong nửa dưới của dãy)**
- **Trả về giá trị biến cờ hiệu**

VD12: Viết hàm tìm 1 phần tử có giá trị gt sử dụng phương pháp tìm kiếm nhị phân.

```
int TimKiemNhiPhan(int a[], int n, int gt)
{
   int dau=0,cuoi=n-1,giua;
   int timthay=-1; //chưa tìm thấy
   while(dau<=cuoi)
   {
        giua=(dau+cuoi)/2;
        if(a[giua]==gt)
        {
             timthay=giua; //tìm thấy tại vị trí giữa
             break;
        }
        if(gt<a[giua])
             cuoi=giua-1;
        else
             dau=giua+1;
   }
   return timthay;
}
```