Một số bài tập lập trình C căn bản

Created by NgoHung

Vấn đề 3: Mảng và con trỏ

Bài toán số 3.1: Nhập xuất và tính tổng các phần tử trong mảng.
Ví dụ: A: 1 5 6 7 4
Kết quả: Tong S = 23
Hướng dẫn: Để giải quyết bài toán cần phải đảm bảo các vấn đề:

- Nhập mảng, có thể xây dựng thành hàm để đóng gói và sử dụng lại. Các thao tác theo yêu cầu:

(1) Nhập số N.
(2) Tiến hành lặp (từ 0 đến N-1) và nhập cho các giá trị A_i trong mảng.
Lưu ý: N phải được vào truyền theo dạng tham biến (tức là &N)

```
void NhapMang1C(int A[],int &N)
{
    printf( "So luong phan tu:" ); scanf( "%d",&N);
    for ( int i=0;i<N;i++)
    {
        printf( "a[%d]=", i );
        scanf( "%d",&(A[i]) );
    }
}
```

- Xuất mảng. Công việc đơn giản là sử dụng vonmgf lặp từ 0 đến N-1 để printf ra các giá trị A_i cho mảng.
Sau đó, printf("\n") để in ký tự xuống dòng.

```
void XuatMang1C( int A[], int N )
{
    printf( "\n Mang :" );
    for ( int i=0;  i < N;  i++)
        printf( "%5d ", A[i] );
    printf( "\n" );
}
```

- Hàm tính tổng các phần tử cho một mảng A
  - Khai báo và khởi tạo tổng S là 0.
  - Sử dụng một vòng lặp (từ 0 đến N-1 để duyệt qua tất cả các giá trị Ai) để tính cộng dồn giá trị của A_i vào tổng S.

```
long TongMang1C(int A[], int N)
{
    long S = 0;
    for ( int i=0; i<N; i++ )
        S = S + A[i] ;
    return S;
}
```

- Viết hàm void main ( ) với nội dung dùng để kiểm tra kết quả thực hiện của hàm.
  - Khai báo mảng A có tối đa 20 phần tử và biến N chỉ số lượng phần tử của A.
  - Gọi hàm nhập mảng để nhập mảng A, với N phần tử.
  - Gọi hàm xuất mảng A, với N phần tử.
  - In giá trị của tổng các phần tử bằng cách truyền trực tiếp giá trị trả về của

việc tính tổng cho printf(……)
Gọi hàm getch( ) trước khi kết thúc hàm main ( ) để dừng lại xem kết quả.

```
void main( )
{
        int A[20], N = 0;
        NhapMang1C( A, N );
        XuatMang1C( A, N );
        printf("\n Tong cac phan tu trong mang %d", TongMang1C(A,
        N) );
        getch();
}
```

Bài toán tương tự:

(1) Tính tổng các số nguyên dương chia hết cho 5.

```
int TongSoChiaHet5(int A[], int N)
{
        int S=0;
        for ( int i=0;i<N; i++ )
                if(A[i] %5==0)
                        S += A[i];
        return S;
}
```

(2) Tính tổng các số nguyên tố trong mảng

```
int LaSoNT(int N)
{
        for ( int i=2;i<N-1; i++ )
                if(N %i == 0)
                        return 0;
        return 1;
}
int TongSoNT(int A[], int N)
{
        int S=0;
        for ( int i=0;i<N; i++ )
                if ( LaSoNT( A[i] ) )
                        S += A[i];
        return S;
}
```

Bài toán số 3.2: Đếm số lần xuất hiện của giá trị X trong mảng A. Đếm số lần xuất hiện của các phần tử trong mảng.

Ví dụ: A: 1 5 6 7 4 1 5 5 1 1

X: 6

Kết quả: So lan xuat hien X la 1

```
So lan xuat hien cua cac phan tu:
        1 ==> 4 5 ==> 3
        6 ==> 1 7 ==> 1
        4 ==> 1 1 ==> 4 …….
```

Hướng dẫn:

+ Viết hàm đếm số lần xuất hiện cuật một giá trị X nào đó được nhập vào, và xem như X nhà là tham số cho việc đếm số lần cuất hiện của nó trong A

```
int DemPtuX(int A[], int N, int X)
{
        int count = 0;
        for (int i=0; i<N; i++)
                if(A[i]==X)
                        count++;
        return count;
}
```

+ Viết hàm in ra số lần xuất hiên của tất cả các phần tử trong mảng, sử dụng lại hàm đã xây dựng ở trước.

```
void InSoLanXH(int A[], int N)
{
        printf( "So lan xuat hien cua cac phan tu trong mang:\n" );
        for (int i=0; i<N; i++)
                printf("%d ==> %d \n", A[i], DemPtuX( A[i] ) );
}
```

+ Xây dựng hàm main để giải quyết bài toán trên gồm:

- Khai báo mảng A, N phần tử.

- Nhập mảng A với N phần tử (lưu ý, phải có định nghĩa hàm nhập /xuất mảng).
- Xuất mảng A với N phần tử.
- Nhập giá trị X cần đếm số là xuất hiện.
- In số lần xuất hiện của X trong A.
- In số lần xuất hiện của các phần tử trong A.

```
void main( )
{
    int A[20], N = 0;
    NhapMang1C( A, N );
    XuatMang1C( A, N );

    printf( "Gia tri X:" ); scanf( "%d", &X );
    printf( "So lan xuat hien cua %d trong A la %d \n", X, DemPtuX(A, N, X) );
    InSoLanXH( A, N );
    getch( );
}
```

Cải tiến: Không in ra các phần tử được lặp lại.

Bài toán số 3.3: Tìm kiếm và thay thế. Tìm kiếm vị trí xuất hiện của x trên mảng A. Thay thế những giá trị A_i là x thành y.

Ví dụ: A: 1 5 6 7 4 1 5 5 1 1
X=5 Y=15
Kết quả: Vi tri xuat hien X la 1
Ket qua thay the: 1 15 6 7 4 1 15 15 1 1

Hướng dẫn:

+ Xây dựng hàm tìm kiếm giá trị X trong mảng A, N phần tử. Sử dụng vòng lặp từ 0 đến N-1 để kiểm tra tất cả các giá trị A_i, nếu bằng x thì trả về vị trí i tìm thấy. Nếu thoát vòng lặp mà không tìm thấy thì trả về là –1.

```
int TimKiem (int A[], int N, int x)
{
    for (int i=0;i<N;i++)
        if (A[i] == x)
            return i; //Tim thay ==> Tra ve vi tri tim
            thay
    return -1; //Khong tim thay
}
```

+ Xây dựng hàm thay thế giá trị x bằng y tại vị trí tìm thấy đầu tiên. Tương tự như tìm kiếm, nhưng khi tìm thấy thì tiến hành gán giá trị mới cho A_i là y.

```
int ThayThe(int A[], int N, int x, int y)
{
    for (int i=0; i<N; i++)
    if(A[i] == x)
    {
        A[i] = y; //Tim thay x ==> thay the thanh y
        return i; //Cham dut qua trinh thay the
    }
    return -1;
}
```

+ Xây dựng hàm thay thế tất cả các giá trị x bằng y tại mỗi vị trí tìm thấy. Sử dụng vòng lặp duyệt qua tất cả các giá trị của A_i, nếu A_i bằng x thì tiến hành gán thành y.

```
void ThayTheTatCa (int A[], int N, int x, int y)
{
    for (int i=0; i<N; i++)
        if(A[i] == x) //Tim thay x ==> thay the thanh y
            A[i] = y;
}
```

Nội dung hàm main( ):

```
void main()
{
    int A[20], N=0, x, y;
    NhapMang1C(A, N); //Ham nhap xuat khong lam lai nua
    XuatMang1C(A, N); // Su du let qua o truoc

    printf("Gia tri x:"); scanf("%d",&x);

    int k = TimKiem (A, N, x);
    if (k >= 0)
```

```
                printf( "Tim thay %d tai vi tri %d trong mang A.\n", x, k
                );
        else
                printf( "Khong tim thay %d trong mang A\n", x );

        printf("gia tri y:"); scanf("%d",&y);

        ThayThe(A, N, x, y);
        printf("Ket qua thay the %d ==> %d dau tien:\n",x, y);
        XuatMang1C(A, N);

        ThayTheTatCa(A, N, x, y);
        printf("Ket qua thay the tat ca %d ==> %d la:\n",x, y);
        XuatMang1C(A, N);

        getch();
}
```

Mở rộng:

+ Tìm kiếm các cặp 2 phần tử gần nhau có tổng chia hết cho 10. Thay thế các phần tử đó bằng tổng của chúng.

Ví dụ: A: 1 19 62 7 8 32 12

Ket qua: 20 20 62 7 40 40 12

```
void ThayTheBangTong(int A[], int N, int x, int y)
{
        for (int i=0; i<N; i++)
                if( (A[i-1]+A[i]) %10 == 0)
                {
                        int k = (A[i-1]+A[i]);
                        A[i-1] = k;
                        A[i] = k;
                }
}
```

Bài toán số 3.4: Kiểm tra mảng có đối xứng hay không? Kiểm tra mảng có tăng dần hay không?

Mảng đối xứng là mảng có phần tử $A_i = A_{N-i-1}$

Nếu mảng không phải là mảng tăng dần, hãy sắp xếp nó thành mảng tăng dần.

Ví dụ: Mảng A: 1 15 6 7 4 7 6 15 1

Kết quả: Mang A doi xung, Mang A khong phai la mang tang dan

Mảng A: 2 5 6 7 14 17 26 26 31

Kết quả: Mang A khong doi xung, Mang A khong phai la mang tang dan

Hướng dẫn:

+ Xây dựng hàm **int KtraDoiXung(A, N)** để kiểm tra tính đối xứng của mảng. Ý tưởng: Giả sử mảng A là mảng đối xứng, sử dụng vòng lặp để tìm kiếm một cặp đối xứng bất kỳ nhưng lại có giá trị không bằng nhau, khi đó trả về là mảng không đối xứng (return 0). Ngược lại là không tìm thấy nên mảng là mảng đối xứng (return 1).

```
int KtraDoiXung (int A[], int N )
{
        for (int i=0; i<N/2; i++)
                if(A[i] != A[N-i-1])
                        return 0; //Cham dut kiem tra, ket qua
                        qua trinh : khong doi xung
        return 1; //Ket qua kiem tra la doi xung
}
```

+ Xây dựng hàm **int KtraMangTang(A, N)** để kiểm tra xem mảng A có phải là mảng tăng hay không. Mảng tăng là mảng có các phần tử đứng sau không nhỏ hơn phần tử đứng trước nó. Ý tưởng: Giả sử mảng A là mảng tăng, sử dụng vòng lặp để kiểm tra có tồn tại phần tử nào nhỏ hơn phần tử đứng trước nó hay không, nếu có thì trả về là mảng không không phải là mảng tăng (return 0). Ngược lại là không tìm thấy nên mảng là mảng tăng (return 1).

```
int KtraMangTang (int A[], int N )
{
        for (int i=1; i<N; i++)
                if(A[i] < A[i-1])
```

```
                return 0; //Cham dut kiem tra, ket qua
                qua trinh : khong tang
        return 1; //Ket qua kiem tra la mang tang
    }
```

+ Xây dựng hàm **void SxepMangTang(A, N)** để sắp xếp mảng A thành mảng tăng dần.
Ý tưởng: Sử dụng 2 vòng lặp lồng nhau để kiểm tra hai phần tử tại vị trí i, j nếu i < j mà A[i] > A[j] thì hoán đổi giá trị của chúng.

```
    void SxepMangTang (int A[], int N )
    {
        for (int i=1; i<N; i++)
            for (int j=1; j<N; j++)
                if (i<j && A[i] > A[j])
                {
                    int k = A[i]; //Tien hanh
                    hoan doi gia tri A[i], A[j]
                    cho nhau
                    A[i] = A[j]; // thong qua
                    bien tam k
                    A[j] = k;
                }
    }
```

+ Xây dựng hàm main( ) cho chương trình để thể hiện kết quả đánh giá trên.

```
    void main()
    {
        int A[20], N=0x, y;
        NhapMang1C(A, N); //Ham nhap xuat khong lam lai nua
        XuatMang1C(A, N); // Su du let qua o truoc

        if (KtraDoiXung (A, N ) )
            printf( " Mang A doi xung.\n");
        else
            printf( " Mang A khong doi xung.\n");

        if (KtraMangTang (A, N ) )
            printf( "Mang A la mang tang \n");
        else
        {
            printf( "Mang A khong phai la mang tang. \n");
            SxepMangTang( A, N );
            printf( "Ket qua sap sep:");
            XuatMang1C(A, N);
        }
        getch();
    }
```

Mở rộng:

+ Kiểm tra mảng A chỉ chứa toàn những số nguyên tố?

+ Kiểm tra mảng giảm dần, Sắp xếp mảng giảm dần.

+ Sắp xếp mảng A có các số dương tăng dần, các số âm giảm dần.

```
                void SxepDuongTangAmGiam (int A[], int N )
                {
                    for (int i=1; i<N; i++)
                        for (int j=1; j<N; j++)
                            if ( (i<j && A[i] > A[j] &&
                            A[i]>0 && A[j]>0)
                               || (i<j && A[i] < A[j]
                            && A[i]<0 && A[j]<0) )
                            {
                                int k =
                                A[i]; //Tien
                                hanh hoan
                                doi gia tri
                                A[i], A[j]
                                A[i] = A[j];
                                // thong
                                qua bien
                                tam k
                                A[j] = k;
```

```
                }
            }
```

+ Kiểm tra mảng A là một chuỗi cấp số cộng có công sai k = 5?

Ví dụ: 1 6 11 16 21 26 31

```
int KtraMangCapSoCong (int A[], int N, int k )
{
    for (int i=1; i<N; i++)
        if (A[i] != A[i-1] + k)
            return 0; //Cham dut, ket qua: khong phai
    return 1; //Ket qua kiem tra la mang cap so cong
}
```

Bài toán số 3.5: Viết thủ tục và chương trình chèn phần tử X vào vị trí k trong mảng A, N phần tử. Xoá phần tử ở vị trí h trong mảng A.

Ví dụ: A : 12 2 3 6 5 17

X = 20 , k = 3 h = 2

Kết quả chèn: 12 2 3 20 6 5 17

Kết quả xoá: 12 2 20 6 5 17

Hướng dẫn:

- Viết hàm chèn một phần tử X vào vị trí k nào đó cho mảng A (có N phần tử). Ý tưởng thuật toán:
  + Dịch chuyển các phần tử từ vị trí k đến N-1 lùi một vị trí, trở thành các phần tử từ vị trí k+1 đến N. Lưu ý, để tránh trường hợp các phần tử đè lên nhau, giải thuật phải tiến hành di dời các phần tử sau trước….đến các phần tử k sau.
  + Gán giá trị cho A[k] là x.
  + Tăng số lượng phần tử của A lên 1, như thế N phải được truyền theo dạng tham biến (tức là &N).

```
void ChenPhanTu(int A[], int &N, int k, int X)
{
    for (int i=N; i>k; i--)
        A[i] = A[i-1];
    A[k] = X;
    N++;
}
```

- Viết hàm xoá một phần tử ở vị trí k trên mảng A (có N phần tử). Ý tưởng thuật toán:
  + Dịch chuyển các phần tử từ vị trí k đến N-1 tiến về trước một vị trí, trở thành các phần tử từ vị trí k-1 đến N-2.
  + Giảm số lượng phần tử của A xuống 1, như thế N phải được truyền theo dạng tham biến (tức là &N).

```
void XoaPhanTu(int A[], int &N, int k)
{
    for (int i=k; i<N-1; i++)
        A[i] = A[i+1];
    N--;
}
```

Nội dung hàm main( ):

```
void main()
{
    int A[20], N=0, x, k, h;
    NhapMang1C(A, N); //Ham nhap xuat khong lam lai nua
    XuatMang1C(A, N); // Su du let qua o truoc

    printf("Gia tri x:"); scanf("%d",&x);
    printf("Vi tri k,h:"); scanf("%d %d",&k,&h);

    printf("Ket qua chen %d vao vi tri %d la:\n", x, k);
    ChenPhanTu(A, N, k, x);
    XuatMang1C(A, N);
    printf("Ket qua xoa phan tu o vi tri %d la:\n", h);
    XoaPhanTu(A, N, h);
    XuatMang1C(A, N);

    getch();
```

```
}
```

Comments

You do not have permission to add comments.