# CÁC THAO TÁC TRÊN MẢNG 1 CHIỀU CƠ BẢN

## Nhập mảng từ bàn phím:

### Phân tích:

Hàm nhập mảng: Tham số là mảng kiểu nguyên, và số phần tử n của mảng sẽ

truyền kiểu tham chiếu (vì muốn sau khi ra khỏi hàm nhập thì giá trị n

cũng sẽ được mang theo đến hàm main để dùng thực hiện các chức năng khác)

### Code minh họa:

void NhapMang(int a[], int& n)

{

    do

    {

        cout << "\nHay nhap so luong phan tu cua mang: ";

        cin >> n;

        if (n < 0 || n > MAX)

        {

            cout << "\nSo luong phan tu khong hop le (1 -> " << MAX << ")!\n";

        }

    } while (n < 0 || n > MAX);

    for (int i = 0; i < n; i++)

    {

        cout << "\nNhap phan tu a[" << i << "] = ";

        cin >> a[i];

    }

}

## Xuất mảng:

### Phân tích:

Hàm xuất mảng: Tham số là mảng kiểu nguyên, và số phần tử n của mảng

truyền kiểu tham chiếu HAY tham trị đều được (vì đối với hàm xuất không

làm thay đổi giá trị của n.

(Nên truyền giá trị n theo kiểu tham trị)

### Code minh họa:

void XuatMang(int a[], int n)

{

    for (int i = 0; i < n; i++)

    {

        cout << a[i] << "\t";

    }

}

## Nhập mảng tự động:

### Code minh họa:

void NhapMangTD(int a[], int& n)

{

    do

    {

        cout << "\nHay nhap so luong phan tu cua mang: ";

        cin >> n;

        if (n < 0 || n > MAX)

        {

            cout << "\nSo luong phan tu khong hop le (1 -> " << MAX << ")!\n";

        }

    } while (n < 0 || n > MAX);

    srand(time(NULL));

    for (int i = 0; i < n; i++)

    {

        a[i] = rand() % (10 - -10 + 1) + -10;

    }

}

## Kiểm tra (Ví dụ mảng có phải toàn số dương không)? ( Tương tự có thể làm các hàm kiểm tra khác,...)

### Code minh họa:

bool KiemTra(int a[], int n)

{

    for (int i = 0; i < n; i++)

    {

        if (a[i] < 0)

            return false; // Nếu có số âm thì trả về kết quả luôn

    }

    return true;

}

## Tìm giá trị lớn nhất

### Code minh họa:

int TimMax(int a[], int n)

{

    // Cho max bằng phần tử đầu tiên của mảng

    int max = a[0];

    for (int i = 0; i < n; i++)

    {

        // Từ đó check max với phần tử thứ 2 của mảng cho đến hết

        if (max < a[i])

            max = a[i];

    }

    return max;

}

## Tìm giá trị nhỏ nhất

### Code minh họa:

int TimMin(int a[], int n)

{

    int min = a[0];

    for (int i = 0; i < n; i++)

    {

        if (min > a[i])

            min = a[i];

    }

    return min;

}

## Sắp xếp mảng tăng dần, giảm dần

### Code minh họa:

void HoanVi(int& a, int& b)

{

    int trungGian;

    trungGian = a;

    a = b;

    b = trungGian;

}

void SapXepMang(int a[], int n, bool isTangDan)

{

    // Thực hiện so sánh theo tuần tự

    for (int i = 0; i < n - 1; i++)

    {

        for (int j = i + 1; j < n; j++)

        {

            // Nếu là sắp xếp tăng

            if (isTangDan)

            {

                if (a[i] > a[j])

                {

                    HoanVi(a[i], a[j]);

                }

            }

            // Nếu là sắp xếp giảm

            else

            {

                if (a[i] < a[j])

                {

                    HoanVi(a[i], a[j]);

                }

            }

        }

    }

}

## Tách mảng ( Ví dụ ở đây tách số âm ra mảng mới)

### Phân tích:

Tách mảng : (Vì số phần tử của [mảng Temp] sau khi tách sẽ thay đổi,

Nên sẽ truyền tham chiếu.

### Code minh họa:

void TachMang(int a[], int n, int Temp[], int& nTemp)

{

    nTemp = 0;

    // Duyệt mảng a để tìm ra các số âm lưu vào trong mảng Temp

    for (int i = 0; i < n; i++)

    {

        if (a[i] < 0)

        {

            Temp[nTemp++] = a[i];

            /\*

            Tương đương với cách viết:

            Temp[nTemp] = a[i];

            nTemp++;

            \*/

        }

    }

}

## Gộp mảng ( Gộp mảng vừa tách vào mảng cũ)

### Code minh họa:

void GopMang(int a[], int n, int Temp[], int nTemp)

{

// Giữ lại số phần tử của mảng a

    int m = n;

   // Gộp số lượng phần tử của Temp[] vào a[]

 n += nTemp;

    for (int i = (n - nTemp); i < n; i++)

    {

        a[i] = Temp[i - m];

    }

/\*

Mảng a[] : 1 2 3 4 5

Temp[] : 6 7

a[5] = Temp[0]

a[6] = Temp[1]

\*/

}

## Tìm kiếm 1 phần tử trong mảng

### Code minh họa:

bool TimKiem(int a[], int n, int x)

{

    for (int i = 0; i < n; i++)

    {

        if (x == a[i])

            return true;

    }

    return false;

}

## Thêm 1 phần tử vào mảng

### Phân tích:

Mảng a ban đầu:

. chỉ số: 0 1 2 3 4 5 6 7

. phần tử: a b c d e f g h

Thêm phần tử X vào vị trí thứ 3 trong mảng.

Mảng a sau khi thêm:

. Chỉ số: 0 1 2 3 4 5 6 7 8

. Phần tử: a b c X d e f g h

VỊ TRÍ SAU = VỊ TRÍ TRƯỚC

a[4] = a[3]

a[5] = a[4]

a[6] = a[5]

a[7] = a[6]

a[8] = a[7]

Nếu for() từ trên xuống thì tất cả giá trị sẽ là của a[3]

(vì qua các lần gán kết qua bị ghi đè)

Do đó, sẽ thực hiện for() từ <<dưới lên>>, rồi thực hiện phép gán giá trị.

Cách 1: Chọn theo VỊ TRÍ SAU, ta thấy:

. a[4] = (Vị trí Thêm + 1) = 3 + 1

. a[8] = Tổng số phần tử ban đầu của mảng a = 8

for (int i = n; i >= n ViTriThem + 1; i --)

{

// TRƯỚC = SAU

a[i] = a[i-1];

}

Cách 2: chọn theo VỊ TRÍ TRƯỚC, ta thấy:

. a[3] = Vị trí Thêm = 3

. a[7] = (Tổng số phần tử ban đầu của mảng a - 1)

= 8 – 1

for (int i = n - 1; i >= ViTriThem; i --)

{

// SAU = TRƯỚC

a[i + 1] = a[i]

}

### Code minh họa:

void ThemPhanTu(int a[], int& n, int phanTuThem, int viTriThem)

{

    for (int i = n; (i >= viTriThem + 1); i--)

    {

        a[i] = a[i - 1];

    }

    // Tăng số lượng phần tử mảng a

    n++;

    a[viTriThem] = phanTuThem;

}

## Xóa 1 phần tử ra khỏi mảng

### Phân tích :

Mảng a ban đầu:

. Chỉ số: 0 1 2 3 4 5 6 7

. Phần tử: a b c d e f g h

Xóa phần tử X tại vị trí thứ 3 trong mảng.

Mảng a sau khi xóa

. chỉ số: 0 1 2 3 4 5 6

. phần tử: a b c e f g h

VỊ TRÍ SAU = VỊ TRÍ TRƯỚC

a[3] = a[4]

a[4] = a[5]

a[5] = a[6]

a[6] = a[7]

Nếu for() từ dưới lên, thì tất cả giá trị sẽ là của a[7]

(vì qua các lần gán kết quả bị ghi đè))

Do đó, sẽ thực hiện for() từ <<<trên xuống>>>, rồi thực hiện phép gán giá trị.

\* Cách 1: Chọn theo VỊ TRÍ SAU, ta thấy:

・a[3] = Vị trí xóa = 3

・a[6] = Ban đầu vị trí cuối mảng là (n - 1), giờ xóa đi 1 phần từ thì vị trí cuối mảng sẽ thành (n - 2)

for (int i = ViTriXoa; i <= (n - 2); i++)

        {

            // TRƯỚC = SAU

            a[i] = a[i + 1];

        }

\* Cách 2: Chọn theo VỊ TRÍ TRƯỚC, ta thấy:

・a[4] = (Vị trí Xóa + 1) = 3 + 1

・a[7] = (Tổng sổ phần tử ban đầu của mảng a - 1) = 8 - 1

 for (int i = viTriXoa + 1; i < n; i++)

        {

            // SAU = TRƯỚC

            a[i - 1] = a[i];

        }

### Code minh họa :

void XoaPhanTu(int a[], int& n, int viTriXoa)

{

    for (int i = viTriXoa + 1; i < n; i++)

    {

        // TRƯỚC = SAU

        a[i - 1] = a[i];

    }

    // Giảm số lượng phần tử của mảng a;

    n--;

}

## Hàm main:

### Code mẫu:

#include <iostream>

#include <conio.h>

#include <cstdlib>

#include <time.h>

using namespace std;

#include "Thuvien.h"

int main()

{

    // Khai báo mảng

    int a[MAX];

    // Khai báo số lượng phẩn tử của mảng

    int n;

    // Nhập

    NhapMangTD(a, n);

    // Xuất

    XuatMang(a, n);

    // Kiểm tra

    bool isToanDuong = KiemTra(a, n);

    if (isToanDuong)

    {

        cout << "\nMang toan so duong.";

    }

    else

    {

        cout << "\nMang co ton tai so am.";

    }

    // Tìm kiếm

    cout << "\nPhan tu nho nhat cua mang la: " << TimMin(a, n);

    cout << "\nPhan tu lon nhat cua mang la: " << TimMax(a, n);

    // Sắp xếp

    cout << "\nMang sap xep tang dan: ";

    SapXepMang(a, n, true);

    XuatMang(a, n);

    cout << "\nMang sap xep giam dan: ";

    SapXepMang(a, n, false);

    XuatMang(a, n);

    // Tách mảng

    int Temp[MAX], nTemp;

    TachMang(a, n, Temp, nTemp);

    cout << "\nMang duoc tach: ";

    XuatMang(Temp, nTemp);

    // Gộp mảng

    GopMang(a, n, Temp, nTemp);

    cout << "\nMang sau khi gop: ";

    XuatMang(a, n);

    // Tìm kiếm

    int x;

    cout << "\nNhap vao so muon tim kiem: ";

    cin >> x;

    bool isResult = TimKiem(a, n, x);

    if (isResult)

    {

        cout << "Tim thay so " << x << " trong mang.";

    }

    else

    {

        cout << "Khong tim thay so " << x << " trong mang.";

    }

    // Thêm phần tử ( Vị trí hợp lệ : 0 -> n)

    int viTriThem, phanTuThem;

    do

    {

        cout << "\nNhap vi tri them (0 -> n): ";

        cin >> viTriThem;

        if (viTriThem < 0 || viTriThem > n)

            cout << "\nVi tri them khong hop le (0 -> n)!";

    } while (viTriThem < 0 || viTriThem > n);

    cout << "\nNhap phan tu them: ";

    cin >> phanTuThem;

    ThemPhanTu(a, n, phanTuThem, viTriThem);

    cout << "\nMang sau khi them phan tu co gia tri = " << phanTuThem << ", vao vi tri: " << viTriThem << "\n";

    XuatMang(a, n);

    // Xóa phần tử ( Vị trí hợp lệ : 0 -> n - 1)

    int viTriXoa;

    do

    {

        cout << "\nNhap vi tri xoa (0 -> n - 1): ";

        cin >> viTriXoa;

        if (viTriXoa < 0 || viTriXoa > n)

            cout << "\nVi tri xoa khong hop le (0 -> n - 1)!";

    } while (viTriXoa < 0 || viTriXoa > n);

    XoaPhanTu(a, n, viTriXoa);

    cout << "\nMang sau khi xoa: \n";

    XuatMang(a, n);

    \_getch();

        return 0;

}