**CÁC THAO TÁC TRÊN MẢNG 1 CHIỀU**

# Xác định phạm vi biến:

**Biến toàn cục** là biến được khai báo ở phân cấp cao hơn vị trí đang xác định. (Phạm vi toàn chương trình)

Ví dụ:

class Program

    {

        static int a = 5;

        static int b = 10;

        static void Main(*string*[] *args*)

        {

            int tong = TinhTong();

            Console.WriteLine(tong);

        }

        static int TinhTong()

        {

            return a + b;

        }

    }

**Như các bạn thấy thì ở đây ta khai báo 2 biến kiểu số nguyên là a = 5, và b = 10**

**Phạm vi biến : Sử dụng trong toàn bộ class Program, nên các hàm main, TinhTong(), .. có thể sử dụng 2 biến này mà không cần truyền tham số.**

**Biến cục bộ**là biến được khai báo ở cùng phân cấp tại vị trí đang xác định. (Phạm vi trong cặp dấu {} )

Ví dụ:

class Program

    {

        static void Main(*string*[] *args*)

        {

            static int a = 5;

            static int b = 10;

            int tong = TinhTong();

            Console.WriteLine(tong);

        }

        static int TinhTong()

        {

            return a + b;

        }

    }

Ở trường hợp này thì chương trình sẽ báo lỗi vì 2 biến kiểu số nguyên a,b được khai báo ở hàm Main, nên phạm vi sử dụng chỉ là ở hàm Main. Hàm TinhTong() không thể sử dụng được.

# Khái niệm về mảng:

Mảng là:

* Tập hợp các đối tượng có cùng kiểu dữ liệu. (Các kiểu cơ sở (int, double, char, string,…) hoặc các kiểu dữ liệu mới do lập trình viên định nghĩa(Sinh viên, Nhân viên,…))
* Mỗi đối tượng trong mảng được gọi là một phần tử.
* Các phần tử phân biệt với nhau bằng chỉ số phần tử. Trong C# chỉ số phần tử là các số nguyên không âm và bắt đầu từ 0 1 2 3…

Đặc điểm của mảng:

* Các phần tử trong mảng dùng chung một tên và được truy xuất thông qua chỉ số phần tử.
* Một mảng cần có giới hạn số phần tử mà mảng có thể chứa.
* Phải cấp phát vùng nhớ mới có thể sử dụng mảng.
* Vị trí ô nhớ của các phần tử trong mảng được cấp phát liền kề nhau.

Tại sao lại dùng mảng:

Để trả lời câu hỏi này ta thử xét trường hợp chúng ta cần chứa họ tên của 10 sinh viên trong trường.

* Nếu sử dụng biến bình thường thì ta cần khai báo 10 biến kiểu string để chứa họ tên của 10 sinh viên. Đến đây vẫn chưa có vấn đề gì lớn!
* Nhưng thực tế thì một trường không phải chỉ có 10 sinh viên, nó có thể là 1000 sinh viên, 10000 sinh viên, 30000 sinh viên. Lúc này vấn đề đã xuất hiện, chúng ta không thể khai báo vài nghìn biến kiểu string được như vậy code sẽ rất dài dòng, rất khó kiểm soát. Khi đó việc sử dụng mảng để lưu trữ là lựa chọn tốt nhất.

Lợi ích của việc sử dụng mảng:

* Gom nhóm các đối tượng có chung tính chất lại với nhau giúp code gọn gàng hơn.
* Để thao tác, dễ quản lý, nâng cấp sửa chữa. Vì lúc này việc thay đổi số lượng sinh viên ta chỉ cần thay đổi số phần tử của mảng là được.
* Dễ dàng áp dụng các cấu trúc lặp vào để xử lý dữ liệu.

# Khai báo, khởi tạo, sử dụng mảng 1 chiều:

## Khai báo mảng 1 chiều:

**<kiểu dữ liệu>** **[]** **<tên mảng>**;

* <kiểu dữ liệu> là kiểu dữ liệu của các phần tử trong mảng.
* Cặp dấu **[]** là ký hiệu cho khai báo mảng 1 chiều.
* <tên mảng> là tên của mảng, cách đặt tên mảng cũng như cách đặt tên biến.

Để sử dụng được mảng ta phải khởi tạo giá trị hoặc cấp phát vùng nhớ cho mảng. Cấp phát vùng nhớ (Sử dụng toán tử new).

Ví dụ:

using System;

namespace ConsoleApp2

{

    class Program

    {

        // Khởi tạo mảng 1 chiều kiểu số nguyên

        static int[] Array = **new** int[100];

        // Khởi tạo biến length thể hiện độ dài sử dụng của mảng

        static int length = 0;

        static void Main(*string*[] *args*)

        {

            Console.ReadKey();

        }

    }

}

Ở đây mình sử dụng biến toàn cục cho việc khởi tạo mảng nhằm mục đích có thể sử dụng cho toàn bộ chương trình.

# Một số thao tác trên mảng 1 chiều:

Mảng là một kiến thức quan trong mình sẽ hướng dẫn các bạn về một số thao tác của mảng nhé.

## Nhập mảng:

Vì ở trên chúng ta đã sử dụng biến toàn cục nên không cần phải truyền tham số cho hàm NhapMang().

public static void NhapMang()

        {

            // B1: Nhập số lượng phần tử của mảng

            Console.Write("Nhap so luong phan tu cua mang: ");

            length = int.Parse(Console.ReadLine());

            // B2: Duyệt mảng -> Nhập dữ liệu

            // Sử dụng vòng lặp để có thể duyệt mảng

            for (int i = 0; i < length; i++)

            {

                Console.Write("Nhap vao a[{0}]= ", i);

                a[i] = int.Parse(Console.ReadLine());

            }

        }

Ngoài có thể sử dụng hàm Random() của C# để tạo một hàm nhập mảng tự động.

public static void NhapMangTuDong()

        {

            // B1: Nhập số lượng phần tử của mảng

            Console.Write("Nhap so luong phan tu cua mang: ");

            length = int.Parse(Console.ReadLine());

            // B2: Sử dụng hàm Randow() để nhập tự động

            Random rd = **new** Random();

            for (int i = 0; i < length; i++)

            {

                a[i] = rd.Next(-10, 10);

            }

        }

## Xuất mảng:

public static void XuatMang()

        {

            // B1: Duyệt -> Xuất kèm định dạng cho đẹp =))

            for (int i = 0; i < length; i++)

            {

                Console.WriteLine("Phan tu thu a[{0}] = {1}", i, a[i]);

            }

        }

## Kiểm tra (Ví dụ kiểm số âm, dương trong mảng, hay kiểm tra những số nguyên tố trong mảng, nói chung có rất nhiều hàm kiểm tra)?

Thì các bước thực hiện, chúng ta sẽ duyệt qua từng phần tử trong mảng sau đó dùng điều kiện để kiểm tra. Nếu đúng thì sẽ trả return về True, ngược lại sai thì return về False.

public bool KiemTra()

{

    for (int i = 0; i < length; i++)

    {

        if (Điều kiện kiểm tra)

            return true; // Nếu đúng với điều kiện kiểm tra

    }

    return false; // ngược lại là không đúng

}

Các bạn có thể áp dụng thay đổi điều kiện để thực hiện các hàm kiểm tra tương tự khác.

## Tìm phần tử lớn nhất trong mảng

public static int TimMax()

{

    // Cho max bằng phần tử đầu tiên của mảng

    int max = a[0];

    for (int i = 0; i < length; 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 phần tử nhỏ nhất trong mảng

public static int TimMin()

{

    // Cho min bằng phần tử đầu tiên của mảng

    int min = a[0];

    for (int i = 0; i < length; i++)

    {

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

        if (min > a[i])

            min = a[i];

    }

    return min;

}

## Thêm 1 phần tử vào một vị trí bất kì

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.

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 >= ViTriThem + 1; i --)

{

// TRƯỚC = SAU

a[i] = a[i-1];

}

Code minh họa:

public static bool KiemTraViTri(int vt)

        {

            // B1: Duyệt kiểm tra vt có tồn tại hay không?

            for (int i = 0; i < length; i++)

            {

                if (vt == i) return true; // Nếu tồn tại trả về True -> Đúng

            }

            return false; // Nếu không tồn tại trả về False -> Sai

        }

public static bool ThemPhanTuVaoViTriBatKi(int vt, int x)

        {

            // B1: Kiểm tra vị trí cần thêm có hợp lệ hay không[0->length+1]?

            bool check = KiemTraViTri(vt);

            if (check) // Nếu có

            { // Thực hiện thêm

                for (int i = length; i >= vt + 1; i--)

                {

                    a[i] = a[i - 1];

                }

                length++;

                a[vt] = x;

                return true; // Thêm thành công

            }

            else

                return false; // Do vị trí không hợp lệ, nên không thêm

        }

## Xóa phần tử tại vị trí bất kì

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.

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];

        }

    n--;

public static bool KiemTraViTri(int vt)

        {

            // B1: Duyệt kiểm tra vt có tồn tại hay không?

            for (int i = 0; i < length; i++)

            {

                if (vt == i) return true; // Nếu tồn tại trả về True -> Đúng

            }

            return false; // Nếu không tồn tại trả về False -> Sai

        }

static public bool XoaTaiViTri(int vt)

        {

            // B1: Kiem tra vi tri xoa do co ton tai hay khong ?

            bool check = KiemTraViTri(vt); // tra ve dung // sai

            if (check) // Neu dung thi thuc hien thuat toan xoa->true

            {

                for (int i = vt; i <= length - 2; i++)

                {

                    a[i] = a[i + 1];

                }

                length--;

                return true;

            }

            else // Neu sai -> false k xoa

                return false;

        }

## Xóa tại vị trí đầu tiên

public static void XoaTaiViTriDauTien()

        {

            XoaTaiViTri(0);

        }

## Xóa tại vị trí cuối cùng

public static void XoaTaiViTriCuoiCung()

        {

            XoaTaiViTri(length - 1);

        }

## Xóa tất cả phần tử trong mảng

public static void XoaTatCacPhanTu()

        {

            length = 0;

        }

Đối với kỹ thuật xóa tất các phần tử thì mình chỉ cần cho length (Chiều dài mảng) = 0.

## Sắp xếp mảng theo chiều tăng

public static void HoanVi(ref int a, ref int b)

        {

            int temp = a;

            a = b;

            b = temp;

        }

public static void SapXepDanhSachTangDan()

        {

            // B1: Chay 2 vong for

            for (int i = 0; i < length - 1; i++)

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

                    if (a[i] > a[j]) // Kiem tra dieu kien sap xep

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

        }

## Sắp xếp theo chiều giảm

public static void HoanVi(ref int a, ref int b)

        {

            int temp = a;

            a = b;

            b = temp;

        }

static public void SapXepDanhSachGiamDan()

        {

            // B1: Chay 2 vong for

            for (int i = 0; i < length - 1; i++)

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

                    if (a[i] < a[j]) // Kiem tra dieu kien sap xep

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

        }

Đối với thuật toán sắp xếp mình dùng thuật toán nổi bọt. Các bạn có thể tham khảo thêm trên mạng nhé.