MẢNG VÀ XÂU KÝ TỰ

Khoa: Công nghệ thông tin 1

Nội dung

1. Khái niệm
2. Khai báo
3. Truy xuất dữ liệu kiểu mảng
4. Một số bài toán trên mảng 1 chiều

Đặt vấn đề

- Ví dụ
  - Chương trình cần lưu trữ 3 số nguyên?

    => Khai báo 3 biến int a1, a2, a3;
  - Chương trình cần lưu trữ 100 số nguyên?

    => Khai báo 100 biến kiểu số nguyên!
  - Người dùng muốn nhập n số nguyên?

    => Không thực hiện được!
- Giải pháp
  - Kiểu dữ liệu mới cho phép lưu trữ một dãy các số nguyên và dễ dàng truy xuất.

Dữ liệu kiểu mảng

- Khái niệm
  - Là một kiểu dữ liệu có cấu trúc do người lập trình định nghĩa.
  - Biểu diễn một dãy các biến có cùng kiểu. Ví dụ: dãy các số nguyên, dãy các ký tự…
  - Kích thước được xác định ngay khi khai báo và không bao giờ thay đổi.
  - NNLT C luôn chỉ định một khối nhớ liên tục cho một biến kiểu mảng.

Khai báo biến mảng

Cú pháp:

```
<kiểu cơ sở> <tên biến mảng>[<số phần tử>];
<kiểu cơ sở> <tên biến mảng>[<N1>][<N2>]…[<Nn>];
```

- <N1>, …, <Nn> : số lượng phần tử của mỗi chiều.

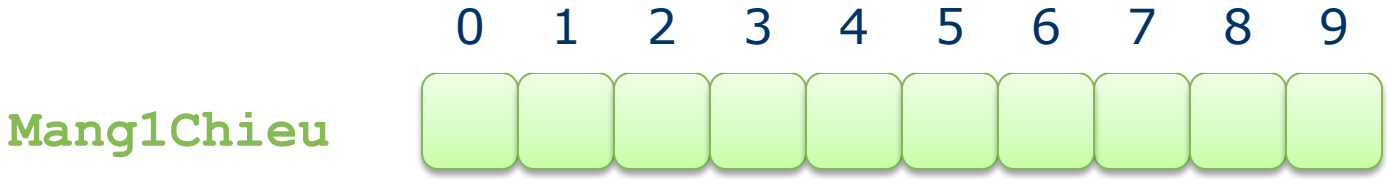
Lưu ý

- Phải xác định <số phần tử> cụ thể (hằng) khi khai báo.
- Mảng nhiều chiều: <tổng số phần tử> = N1*N2*…*Nn
- Bộ nhớ sử dụng = <tổng số phần tử>*sizeof(<kiểu cơ sở>)
- Bộ nhớ sử dụng phải ít hơn 64KB (65535 Bytes)
- Một dãy liên tục có chỉ số từ 0 đến <tổng số phần tử>-1

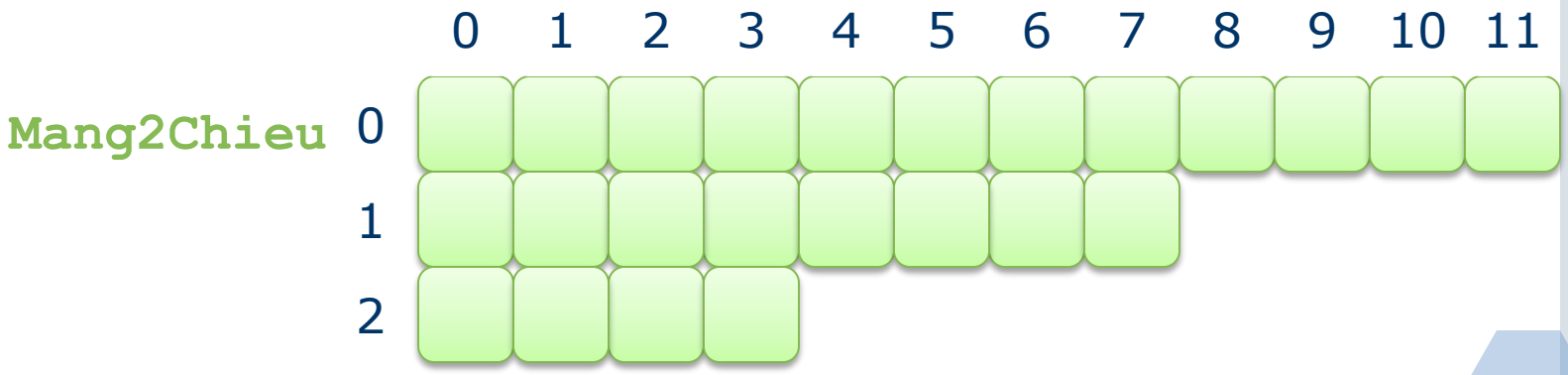
Khai báo biến mảng

Ví dụ

```
int Mang1Chieu[10];
```




```
int Mang2Chieu[3][4];
```



Số phần tử của mảng

- Phải xác định cụ thể số phần tử ngay lúc khai báo, không được sử dụng biến hoặc hằng thường

```
int n1 = 10; int a[n1];
const int n2 = 20; int b[n2];
```

- Nên sử dụng chỉ thị tiền xử lý #define để định nghĩa số phần tử mảng

```
#define n1 10
#define n2 20
int a[n1];          // ⇔ int a[10];
int b[n1][n2];      // ⇔ int b[10][20];
```

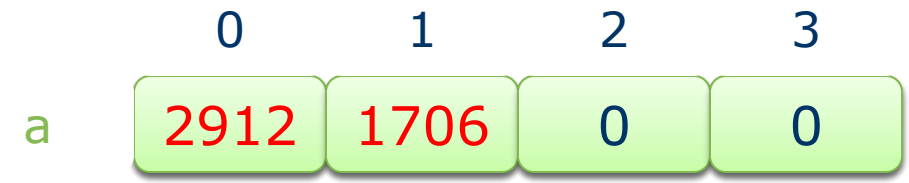
Khởi tạo giá trị cho mảng lúc khai báo

- Gồm các cách sau
  - Khởi tạo giá trị cho mọi phần tử của mảng

```
int a[4] = {2912, 1706, 1506, 1904};
```

  - Khởi tạo giá trị cho một số phần tử đầu mảng

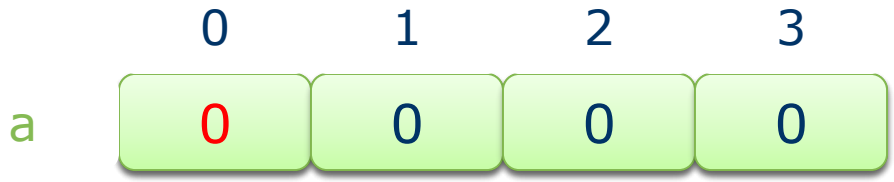
```
int a[4] = {2912, 1706};
```



Khởi tạo giá trị cho mảng lúc khai báo

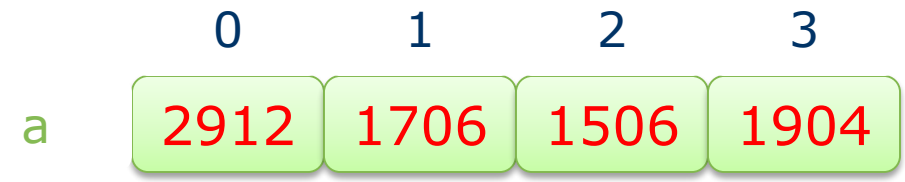
- Gồm các cách sau
  - Khởi tạo giá trị 0 cho mọi phần tử của mảng

```
int a[4] = {0};
```



  - Tự động xác định số lượng phần tử

```
int a[] = {2912, 1706, 1506, 1904};
```



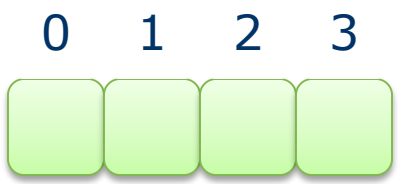
Truy xuất đến một phần tử

- Thông qua chỉ số

```
<tên biến mảng>[<gt cs1>][<gt cs2>]…[<gt csn>]
```

- Ví dụ
  - Cho mảng như sau

```
int a[4];
```



  - Các truy xuất
    - Hợp lệ: a[0], a[1], a[2], a[3]
    - Không hợp lệ: a[-1], a[4], a[5], …

➔ Cho kết quả thường không như mong muốn!

Lấy địa chỉ của phần tử mảng

- Sử dụng toán tử &

```
&<tên biến mảng>[i]   (i là chỉ số của mảng)
```

- Chú ý:
  - Tên của mảng chứa địa chỉ đầu của mảng.
  - Ví dụ: Có int a[10] thì a = &a[0]

Gán dữ liệu kiểu mảng

- Không được sử dụng phép gán thông thường mà phải gán trực tiếp giữa các phần tử tương ứng

```
<biến mảng đích> = <biến mảng nguồn>; //sai
<biến mảng đích>[<chỉ số thứ i>] := <giá trị>;
```

- Ví dụ

```
#define MAX 3
typedef int MangSo[MAX];
MangSo a = {1, 2, 3}, b;

b = a;        // Sai
for (int i = 0; i < 3; i++) b[i] = a[i];
```

Một số lỗi thường gặp

- Khai báo không chỉ rõ số lượng phần tử
  - int a[]; => int a[100];
- Số lượng phần tử liên quan đến biến hoặc hằng
  - int n1 = 10; int a[n1]; => int a[10];
  - const int n2 = 10; int a[n2]; => int a[10];
- Khởi tạo cách biệt với khai báo
  - int a[4]; a = {2912, 1706, 1506, 1904};
    => int a[4] = {2912, 1706, 1506, 1904};
- Chỉ số mảng không hợp lệ
  - int a[4];
  - a[-1] = 1; a[10] = 0;

Truyền mảng cho hàm

Truyền mảng cho hàm

- Tham số kiểu mảng trong khai báo hàm giống như khai báo biến mảng

```
void SapXepTang(int a[100]);
```

- Tham số kiểu mảng truyền cho hàm chính là địa chỉ của phần tử đầu tiên của mảng
  - Có thể bỏ số lượng phần tử hoặc sử dụng con trỏ.
  - Mảng có thể thay đổi nội dung sau khi thực hiện hàm.

```
void SapXepTang(int a[]);
void SapXepTang(int *a);
```

Truyền mảng cho hàm

- Truyền mảng cho hàm
  - Số lượng phần tử thực sự truyền qua biến khác

```
void SapXepTang(int a[100], int n);
void SapXepTang(int a[], int n);
void SapXepTang(int *a, int n);
```

- Lời gọi hàm

```
void NhapMang(int a[], int &n);
void XuatMang(int a[], int n);
void main()
{
        int a[100], n;
        NhapMang(a, n);
        XuatMang(a, n);
}
```

Một số bài toán cơ bản

- Viết hàm thực hiện từng yêu cầu sau
  - Nhập mảng
  - Xuất mảng
  - Tìm kiếm một phần tử trong mảng
  - Kiểm tra tính chất của mảng
  - Tách mảng / Gộp mảng
  - Tìm giá trị nhỏ nhất/lớn nhất của mảng
  - Sắp xếp mảng giảm dần/tăng dần
  - Thêm/Xóa/Sửa một phần tử vào mảng

Một số quy ước

- Số lượng phần tử

```
#define MAX 100
```

- Các hàm
  - Hàm void HoanVi(int &x, int &y): hoán vị giá trị của hai số nguyên.
  - Hàm int LaSNT(int n): kiểm tra một số có phải là số nguyên tố. Trả về 1 nếu n là số nguyên tố, ngược lại trả về 0.

Hàm HoanVi & Hàm LaSNT

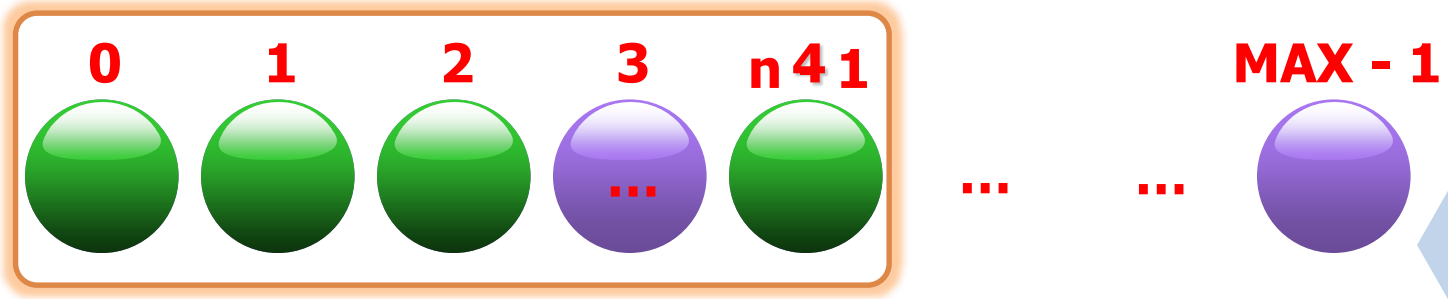
```
void HoanVi(int &x, int &y)
{
	int tam = x; x = y; y = tam;
}

int LaSNT(int n)
{
	int i, dem = 0;
	for (i = 1; i <= n; i++)
		if (n%i == 0)
			dem++;

	if (dem == 2)
		return 1;
	else return 0;
}
```

Nhập mảng

- Yêu cầu
  - Cho phép nhập mảng a, số lượng phần tử n
- Ý tưởng
  - Cho trước một mảng có số lượng phần tử là MAX.
  - Nhập số lượng phần tử thực sự n của mảng.
  - Nhập từng phần tử cho mảng từ chỉ số 0 đến n – 1.



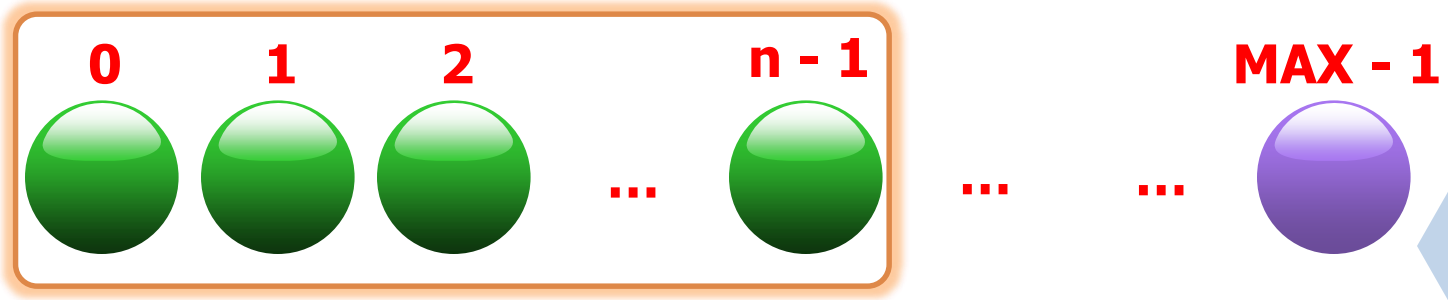
Hàm Nhập Mảng

```
void NhapMang(int a[], int &n)
{
        printf("Nhap so luong phan tu n: ");
        scanf("%d", &n);

        for (int i = 0; i < n; i++)
        {
                printf("Nhap phan tu thu %d: ", i);
                scanf("%d", &a[i]);
        }
}
```

Xuất mảng

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử n. Hãy xuất nội dung mảng a ra màn hình.
- Ý tưởng
  - Xuất giá trị từng phần tử của mảng từ chỉ số 1 đến n.



Hàm Xuất Mảng

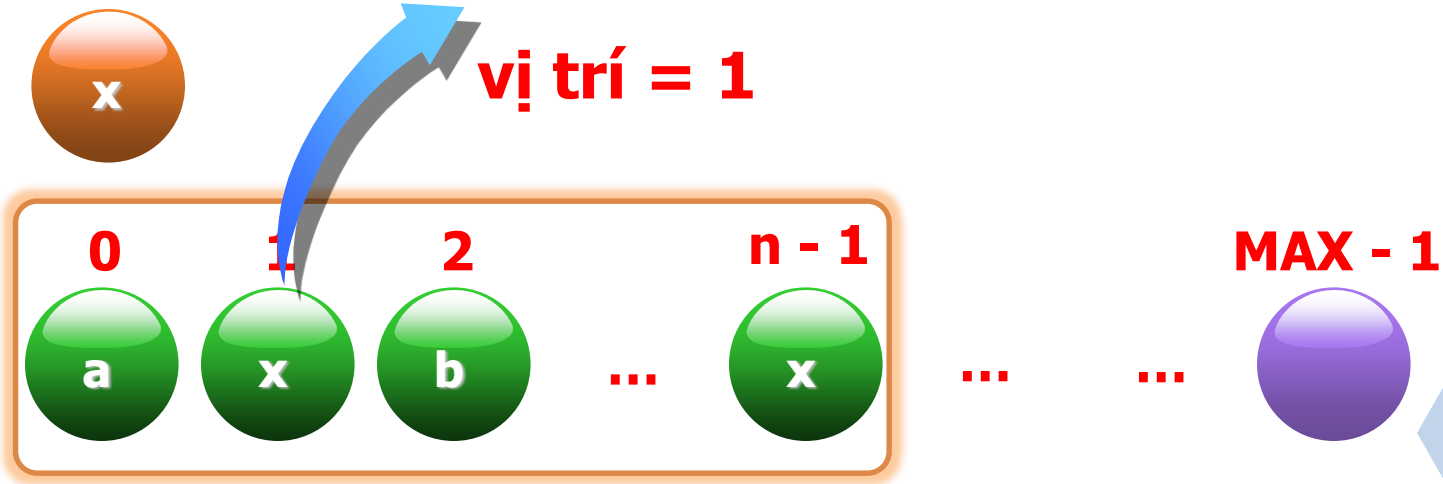
```
void XuatMang(int a[], int n)
{
        printf("Noi dung cua mang la: ");

        for (int i = 0; i < n; i++)
                printf("%d ", a[i]);

        printf("\n");
}
```

Tìm kiếm một phần tử trong mảng

- Yêu cầu
  - Tìm xem phần tử x có nằm trong mảng a kích thước n hay không? Nếu có thì nó nằm ở vị trí đầu tiên nào.
- Ý tưởng
  - Xét từng phần của mảng a. Nếu phần tử đang xét bằng x thì trả về vị trí đó. Nếu kô tìm được thì trả về -1.



Hàm Tìm Kiếm (dùng while)

```
int TimKiem(int a[], int n, int x)
{
        int vt = 0;

        while (vt < n && a[vt] != x)
                vt++;

        if (vt < n)
                return vt;
        else
                return -1;
}
```

Hàm Tìm Kiếm (dùng for)

```
int TimKiem(int a[], int n, int x)
{
        for (int vt = 0; vt < n; vt++)
                if (a[vt] == x)
                        return vt;

        return -1;
}
```

Kiểm tra tính chất của mảng

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử n. Mảng a có phải là mảng toàn các số nguyên tố hay không?
- Ý tưởng
  - Cách 1: Đếm số lượng số ngtố của mảng. Nếu số lượng này bằng đúng n thì mảng toàn ngtố.
  - Cách 2: Đếm số lượng số không phải ngtố của mảng. Nếu số lượng này bằng 0 thì mảng toàn ngtố.
  - Cách 3: Tìm xem có phần tử nào không phải số ngtố không. Nếu có thì mảng không toàn số ngtố.

Hàm Kiểm Tra (Cách 1)

```
int KiemTra_C1(int a[], int n)
{
        int dem = 0;

        for (int i = 0; i < n; i++)
              if (LaSNT(a[i]) == 1) // có thể bỏ == 1
                    dem++;

        if (dem == n)
              return 1;
        return 0;
}
```

Hàm Kiểm Tra (Cách 2)

```
int KiemTra_C2(int a[], int n)
{
        int dem = 0;

        for (int i = 0; i < n; i++)
              if (LaSNT(a[i]) == 0) // Có thể sử dụng !
                    dem++;

        if (dem == 0)
              return 1;
        return 0;
}
```

Hàm Kiểm Tra (Cách 3)

```
int KiemTra_C3(int a[], int n)
{
        for (int i = 0; i < n ; i++)
              if (LaSNT(a[i]) == 0)
                    return 0;

        return 1;
}
```

Tách các phần tử thỏa điều kiện

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử na. Tách các số nguyên tố có trong mảng a vào mảng b.
- Ý tưởng
  - Duyệt từ phần tử của mảng a, nếu đó là số nguyên tố thì đưa vào mảng b.

Hàm Tách Số Nguyên Tố

```
void TachSNT(int a[], int na, int b[], int &nb)
{
        nb = 0;

        for (int i = 0; i < na; i++)
                if (LaSNT(a[i]) == 1)
                {
                        b[nb] = a[i];
                        nb++;
                }
}
```

Tách mảng thành 2 mảng con

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử na. Tách mảng a thành 2 mảng b (chứa số nguyên tố) và mảng c (các số còn lại).
- Ý tưởng
  - Cách 1: viết 1 hàm tách các số nguyên tố từ mảng a sang mảng b và 1 hàm tách các số không phải nguyên tố từ mảng a sang mảng c.
  - Cách 2: Duyệt từ phần tử của mảng a, nếu đó là số nguyên tố thì đưa vào mảng b, ngược lại đưa vào mảng c.

Hàm Tách 2 Mảng

```
void TachSNT2(int a[], int na,
              int b[], int &nb, int c[], int &nc)
{
      nb = 0;
      nc = 0;

      for (int i = 0; i < na; i++)
            if (LaSNT(a[i]) == 1)
            {
                  b[nb] = a[i]; nb++;
            }
            else
            {
                  c[nc] = a[i]; nc++;
            }
}
```

Gộp 2 mảng thành một mảng

- Yêu cầu
  - Cho trước mảng a, số lượng phần tử na và mảng b số lượng phần tử nb. Gộp 2 mảng trên theo tứ tự đó thành mảng c, số lượng phần tử nc.
- Ý tưởng
  - Chuyển các phần tử của mảng a sang mảng c

    => nc = na
  - Tiếp tục đưa các phần tử của mảng b sang mảng c

    => nc = nc + nb

Hàm Gộp Mảng

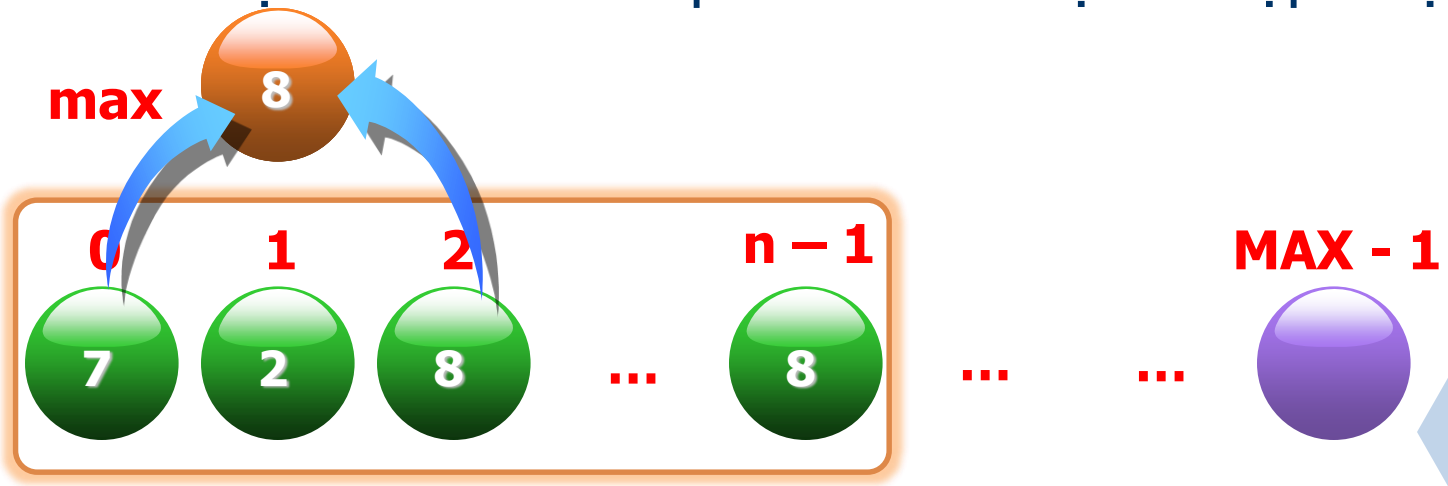
```
void GopMang(int a[], int na, int b[], int nb,
             int c[], int &nc)
{
      nc = 0;

      for (int i = 0; i < na; i++)
      {
            c[nc] = a[i]; nc++; // c[nc++] = a[i];
      }

      for (int i = 0; i < nb; i++)
      {
            c[nc] = b[i]; nc++; // c[nc++] = b[i];
      }
}
```

Tìm giá trị lớn nhất của mảng

- Yêu cầu
  - Cho trước mảng a có n phần tử. Tìm giá trị lớn nhất trong a (gọi là max)
- Ý tưởng
  - Giả sử giá trị max hiện tại là giá trị phần tử đầu tiên a[0]
  - Lần lượt kiểm tra các phần tử còn lại để cập nhật max.



Hàm tìm Max

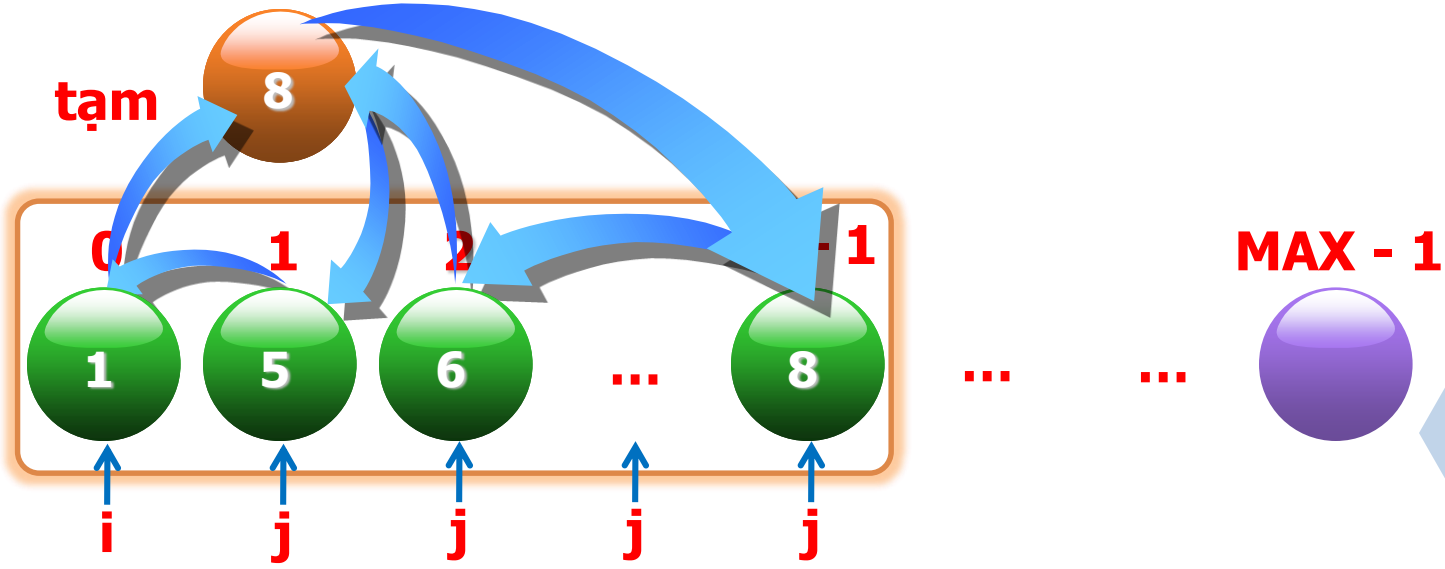
```
int TimMax(int a[], int n)
{
        int max = a[0];

        for (int i = 1; i < n; i++)
              if (a[i] > max)
                    max = a[i];

        return max;
}
```

Sắp xếp mảng thành tăng dần

- Yêu cầu
  - Cho trước mảng a kích thước n. Hãy sắp xếp mảng a đó sao cho các phần tử có giá trị tăng dần.
- Ý tưởng
  - Sử dụng 2 biến i và j để so sánh tất cả cặp phần tử với nhau và hoán vị các cặp nghịch thế (sai thứ tự).



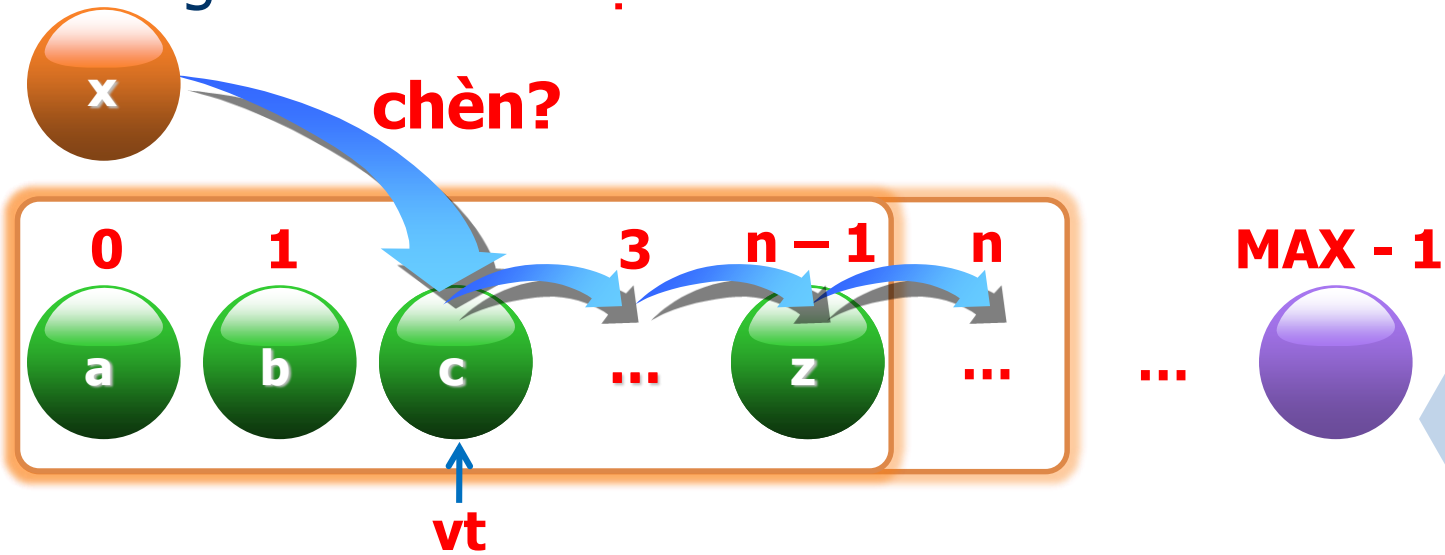
Hàm Sắp Xếp Tăng

```
void SapXepTang(int a[], int n)
{
	int i, j;

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

Thêm một phần tử vào mảng

- Yêu cầu
  - Thêm phần tử x vào mảng a kích thước n tại vị trí vt.
- Ý tưởng
  - "Đẩy" các phần tử bắt đầu tại vị trí vt sang phải 1 vị trí.
  - Đưa x vào vị trí vt trong mảng.
  - Tăng n lên 1 đơn vị.



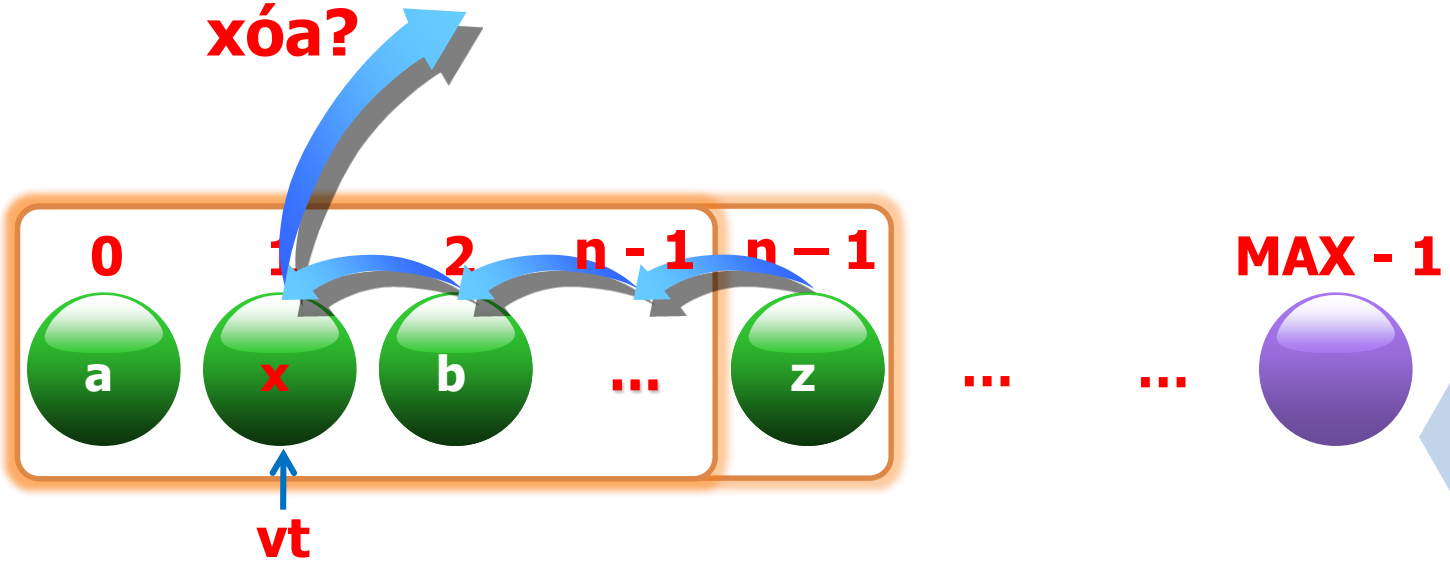
Hàm Thêm

```
void Them(int a[], int &n, int vt, int x)
{
        if (vt >= 0 && vt <= n)
        {
                for (int i = n; i > vt; i--)
                        a[i] = a[i - 1];

                a[vt] = x;
                n++;
        }
}
```

Xóa một phần tử trong mảng

- Yêu cầu
  - Xóa một phần tử trong mảng a kích thước n tại vị trí vt
- Ý tưởng
  - "Kéo" các phần tử bên phải vị trí vt sang trái 1 vị trí.
  - Giảm n xuống 1 đơn vị.



Hàm Xóa

```
void Xoa(int a[], int &n, int vt)
{
        if (vt >= 0 && vt < n)
        {
                for (int i = vt; i < n - 1; i++)
                        a[i] = a[i + 1];

                n--;
        }
}
```

Mảng 2 chiều

- Nhập, xuất
- Tính tổng, tích
- Tìm kiếm
- Đếm
- Sắp xếp
- Thêm, xoá, thay thế

Khai báo

- Cách 1: Con trỏ hằng

<Kiểu dữ liệu> <Tên mảng> [<Số dòng tối đa>][<Số cột tối đa>];

- Ví dụ:

int A[10][10]; // Khai báo mảng 2 chiều kiểu int
//gồm 10 dòng, 10 cột

float b[10][10]; // Khai báo mảng 2 chiều kiểu
//float gồm 10 dòng, 10 cột

Truy xuất phần tử của mảng

- Để truy xuất các thành phần của mảng hai chiều ta phải dựa vào chỉ số dòng và chỉ số cột.
- Ví dụ:

  int A[3][4] = { {2,3,9,4} , {5,6,7,6} , {2,9,4,7} };

  Với các khai báo như trên ta có :

  A[0][0] = 2; A[0][1] = 3;

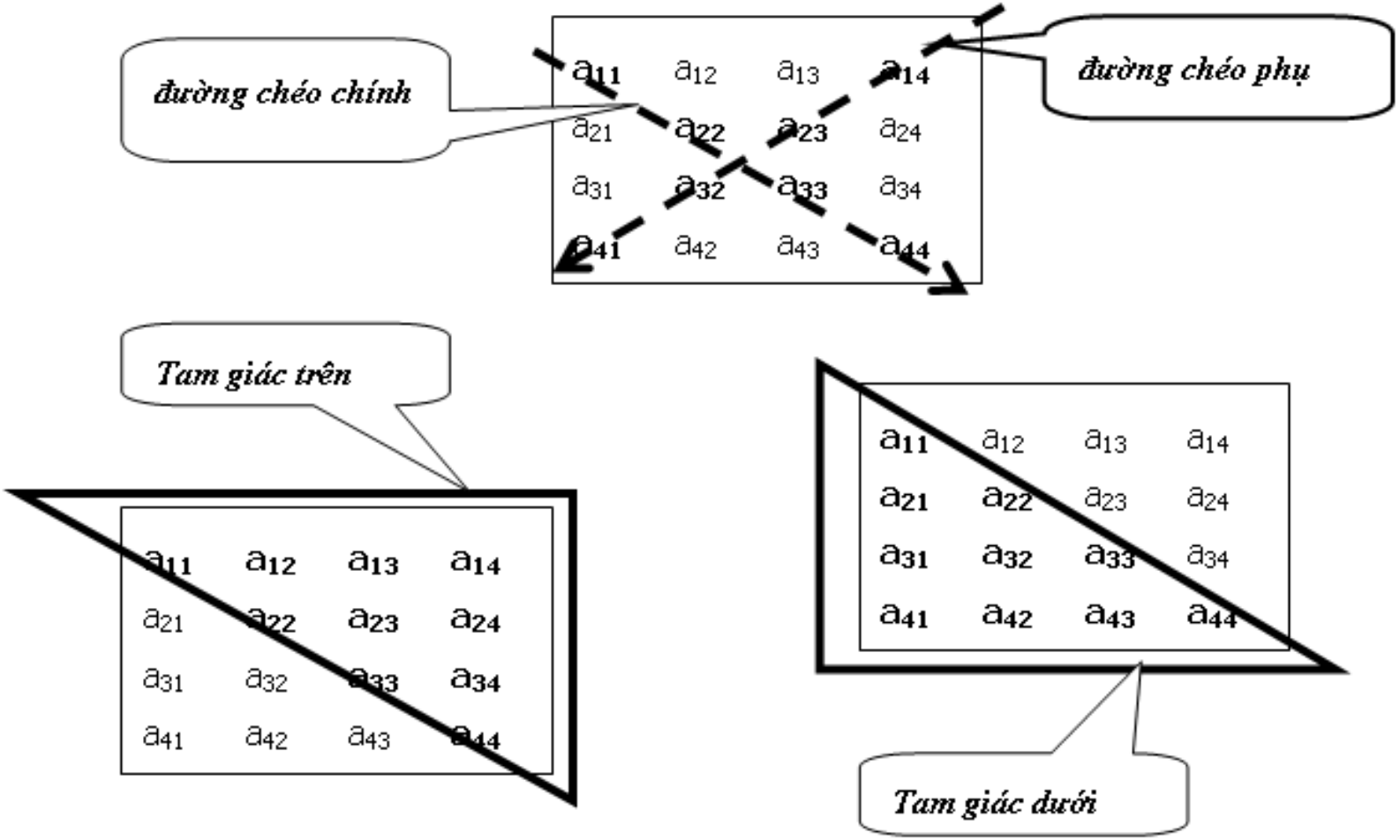
  A[1][1] = 6; A[1][3] = 6;

- *Khi nhập liệu cho mảng hai chiều, nếu là mảng các số nguyên thì ta nhập liệu theo cách thông thường. Nhưng nếu là mảng các số thực thì ta phải thông qua biến trung gian.*

Ma trận vuông và các khái niệm liên quan

- Đường chéo chính
- Đường chéo phụ
- Tam giác trên
- Tam giác dưới

Ma trận vuông và các khái niệm liên quan



Một số bài toán cơ bản

- Viết hàm thực hiện từng yêu cầu sau
  - Nhập mảng
  - Xuất mảng
  - Xuất đường chéo chính/phụ/tam giác trên/tam giác dưới
  - Tính tổng
  - Đếm
  - Tìm kiếm một phần tử trong mảng
  - Tìm giá trị nhỏ nhất/lớn nhất của mảng
  - Kiểm tra tính chất của mảng

Một số bài toán cơ bản

- Nhập, xuất: Viết các hàm
  - Nhập ma trận kích thước nxm các số nguyên dương (3<n, m<10)
  - Tạo ma trận kích thước nxm (3<n, m<10) các số nguyên dương có giá trị ngẫu nhiên trong khoảng -10 đến 20.
  - Xuất ma trận ra màn hình.
  - Viết hàm tính tổng các phần tử chẵn trong ma trận
  - Viết hàm tính tổng các phần tử trên cùng một dòng.

Chuỗi ký tự – Strings

- **Một số qui tắc**
- **Nhập / xuất**
- **Con trỏ và chuỗi ký tự**
- **Một số hàm thư viện**

Chuỗi ký tự - **Một số qui tắc**

- Chuỗi có thể được định nghĩa như là một mảng kiểu ký tự, được kết thúc bằng ký tự null
- Mỗi ký tự trong chuỗi chiếm một byte và ký tự cuối cùng của chuỗi là "\0" (null)
- Được khai báo và truyền tham số như mảng một chiều.

```
char           s[100];
unsigned char  s1[1000];
```

Chuỗi ký tự – Khai báo

- Cách 1: Con trỏ hằng

  char < Tên chuỗi > [< Số ký tự tối đa của chuỗi >];

  - Ví dụ: *char chuoi[25];*
  - Ý nghĩa: khai báo 1 mảng kiểu ký tự tên là chuoi có 25 phần tử (như vậy tối đa ta có thể nhập 24 ký tự vì phần tử thứ 25 đã chứa ký tự kết thúc chuỗi '\0')

```
char     s[100];
unsigned char s1[1000];
```

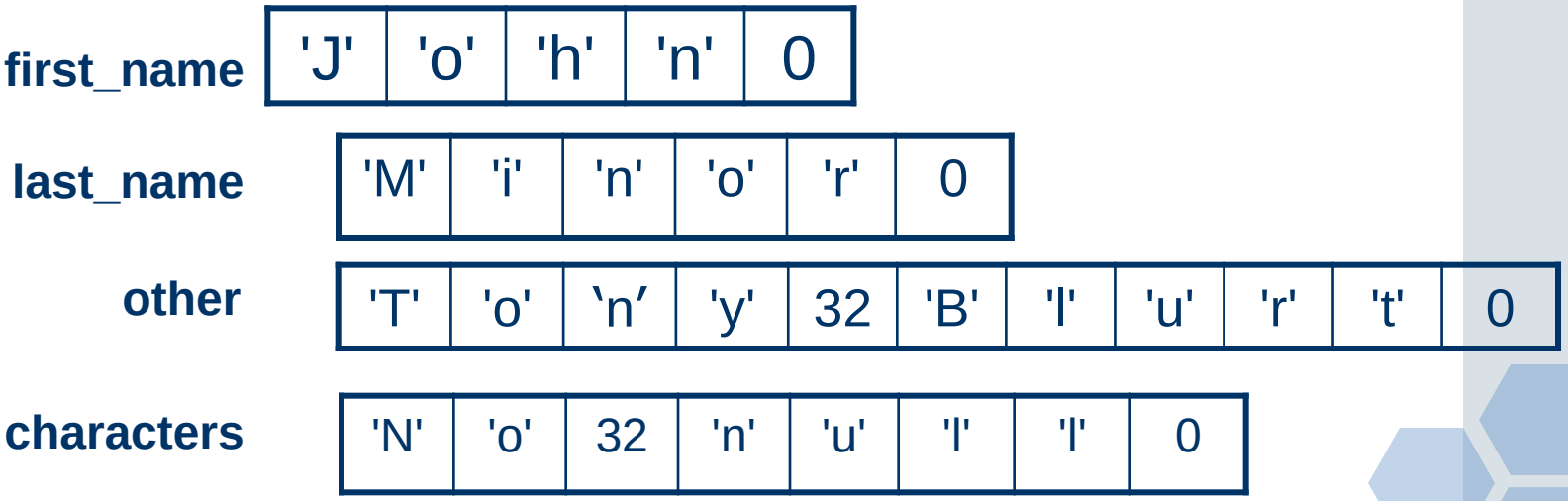
Chuỗi ký tự - Ví dụ

```
char  first_name[5] = { 'J', 'o', 'h', 'n', '\0' };

char  last_name[6]  = "Minor";

char  other[]       = "Tony Blurt";

char  characters[7] = "No null";
```



Chuỗi ký tự - Nhập / xuất

- Có thể nhập / xuất chuỗi ký tự s bằng cách nhập từng ký tự của s
- Hoặc sử dụng các hàm `scanf` và `printf` với ký tự định dạng "`%s`"

```
char  other[] = "Tony Blurt";
printf("%s\n", other);
```

- Nhập chuỗi có khoảng trắng dùng hàm **gets**

```
char  name[100];
printf("Nhap mot chuoi ky tu: ");
gets(name);
```

Chuỗi ký tự - Nhập

- Nhập chuỗi

char *gets(char *s);

- Ví dụ:

```
void main()
{
    char chuoi[80];
    printf("Nhap vao chuoi:");
    gets(chuoi);
    printf("Chuoi vua nhap la: %s\n", chuoi);
}
```

Chuỗi ký tự - Xuất

- Xuất chuỗi

  int puts(const char *s);

- Ví dụ:

```
void main()
{
        char chuoi[] = "Dai hoc Nha Trang\n";
        puts(chuoi);
}
```

- **Lưu ý:** Cách truy xuất các ký tự tương tự như mảng một chiều.

Ví dụ

Nhập vào một chuỗi ký tự, xuất ra màn hình chuỗi bị đảo ngược thứ tự các ký tự.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void DaoChuoi(char s1[], char s2[])
{
    int l=strlen(s1);
    for(int i=0; i<l; i++)
    s2[i]=s1[l-i-1];
    s2[i]='\0';
}
void main()
{
    char s1[100], s2[100];
    clrscr();
    printf("\nNhap vao chuoi ky tu: ");
    gets(s1);
    DaoChuoi(s1, s2);
    printf("\nKet qua sau khi dao nguoc
        chuoi: %s", s2);
}
```

Lưu ý: kết thúc chuỗi

```
#include <stdio.h>

int  main()
{
     char other[] = "Tony Blurt";

     printf("%s\n", other);

     other[4] = '\0';

     printf("%s\n", other);

     return 0;
}
```

"Blurt" sẽ không được in ra

```
Tony Blurt
Tony
```

other

'T'	'o'	'n'	'y'	32	'B'	'l'	'u'	'r'	't'	0

Chuỗi ký tự – Một số hàm thư viện

Các hàm xử lý chuỗi được tìm thấy trong thư viện chuẩn <string.h>

- **Lấy độ dài chuỗi**

```
l = strlen(s);
```

- **Đổi toàn bộ các ký tự của chuỗi thành IN HOA**

```
strupr(s);
```

- **Đổi toàn bộ các ký tự của chuỗi thành in thường**

```
strlwr(s);
```

Chuỗi ký tự – Một số hàm thư viện

- **So sánh chuỗi: so sánh theo thứ tự từ điển**

Phân biệt IN HOA – in thường:

```
int strcmp(const char *s1, const char *s2);
```

Không phân biệt IN HOA – in thường:

```
int stricmp(const char *s1, const char *s2);
```

Chuỗi ký tự – ví dụ strcmp

```
#include <stdio.h>

int main()
{
    char  s1[] = "Minor";
    char  s2[] = "Tony";
    int cmp = strcmp(s1, s2);
    if (cmp < 0)
        printf("%s < %s", s1, s2);
    else
        if (cmp == 0)
            printf("%s = %s", s1, s2);
        else
            printf("%s > %s", s1, s2);
    return 0;
}
```

```
Minor < Tony
```

Chuỗi ký tự – Một số hàm thư viện

- **Gán nội dung chuỗi:**
  - ***Chép toàn bộ chuỗi source sang chuỗi dest:***

```
int strcpy(char *dest, const char *src);
```

  - ***Chép tối đa n ký tự từ source sang dest:***

```
int strncpy(char *dest,
            const char *src, int n);
```

- **Tạo chuỗi mới từ chuỗi đã có:**

```
char *strdup(const char *src);
```

Chuỗi ký tự – ví dụ strcpy

```
#include <stdio.h>

int main()
{
    char s[] = "Tony Blurt";
    char s2[100], *s3;

    strcpy(s2, s);
    printf("%s\n", s2);
    strncpy(s2 + 2, "12345", 3);
    printf("%s\n", s2);
    s3 = strdup(s + 5);
    printf("%s\n", s3);
    free(s3);
    return 0;
}
```

```
Tony Blurt
To123Blurt
Blurt
```

Chuỗi ký tự – Một số hàm thư viện

- **Nối chuỗi:**

```
char *strcat(char *dest,const char *src);
```

- **Tách chuỗi:**

```
char *strtok(char *s,const char *sep);
```

Trả về địa chỉ của đoạn đầu tiên. Muốn tách đoạn kế tiếp tham số thứ nhất sẽ là NULL

Chuỗi ký tự – **ví dụ** strtok

```
#include <stdio.h>

#define SEPARATOR "., "

int main()
{
    char s[]= "Thu strtok: 9,123.45";
    char *p;

    p = strtok(s, SEPARATOR);
    while (p != NULL)   {
        printf("%s\n", p);
        p = strtok(NULL, SEPARATOR);
    }
    return 0;
}
```

```
Thu
strtok:
9
123
45
```

Chuỗi ký tự – Một số hàm thư viện

- **Tìm một ký tự trên chuỗi:**

```
char *strchr(const char *s, int c);
```

- **Tìm một đoạn ký tự trên chuỗi:**

```
char *strstr(const char *s1,const char
*s2);
```

Chuỗi ký tự – ví dụ tìm kiếm

```
#include <stdio.h>

int main()
{
    char s[]= "Thu tim kiem chuoi";
    char *p;

    p = strchr(s, 'm');
    printf("%s\n", p);
    p = strstr(s, "em");
    printf("%s\n", p);
    return 0;
}
```

```
m kiem chuoi
em chuoi
```

Chuỗi ký tự – chèn một đoạn ký tự

```
#include <stdio.h>

void StrIns(char *s, char *sub)
{
    int  len = strlen(sub);
    memmove(s + len, s, strlen(s)+1);
    strncpy(s, sub, len);
}
int main()
{
    char s[]= "Thu chen";

    StrIns(s, "123");       printf("%s\n", s);
    StrIns(s + 8, "45");    printf("%s\n", p);
    return 0;
}
```

```
123 Thu chen
123 Thu 45chen
```

Chuỗi ký tự – xóa một đoạn ký tự

```
#include <stdio.h>

void StrDel(char *s, int n)
{
    memmove(s, s + n, strlen(s+n)+1);
}
int main()
{
    char s[]= "Thu xoa 12345";

    StrDel(s, 4);        printf("%s\n", s);
    StrDel(s + 4, 3);    printf("%s\n", p);
    return 0;
}
```

```
xoa 12345
xoa 45
```