



TRƯỜNG ĐẠI HỌC THỦY LỢI  
Khoa Công nghệ thông tin  
Bộ môn Tin học và KTTT

# NHẬP MÔN LẬP TRÌNH

## INTRODUCTION TO COMPUTER PROGRAMMING

Giảng viên: TS.GVC Bùi Thị Thanh Xuân

Email: xuanbtt@tlu.edu.vn

Điện thoại: 0902001581



# Chương 4: Mảng, con trỏ và xâu ký tự



## 1. Mảng

- Khái niệm
- Khai báo và sử dụng
- Các thao tác thường gặp

## 2. Con trỏ

- Khái niệm và cách khai báo
- Toán tử địa chỉ (&), toán tử nội dung (\*)
- Phép toán trên con trỏ
- Con trỏ và mảng

## 3. Xâu ký tự

- Khái niệm, khai báo và sử dụng
- Các hàm xử lý ký tự và xâu ký tự
- Mảng xâu ký tự



# Khái niệm mảng



- Kiểu mảng là một kiểu dữ liệu gồm
  - Một số hữu hạn thành phần.
  - Các thành phần có cùng một kiểu: kiểu cơ sở hay là kiểu thành phần.
- Mỗi phần tử của mảng được tham khảo thông qua
  - Tên mảng và
  - Chỉ số của phần tử trong mảng



# Khai báo mảng



**Kiểu\_dữ\_liệu Tên\_Mảng[Kích\_thước];**

- **Kiểu\_dữ\_liệu:** kiểu của các phần tử trong mảng (nguyên, thực, ký tự, chuỗi, mảng,...)
- **Tên\_Mảng:** tên của mảng khai báo
- **Kích\_thước:** số phần tử (tối đa) trong mảng.

**Ví dụ:**

// khai báo mảng 50 phần tử có kiểu dữ liệu int

int DiemTin[50];

//Mảng tên A gồm 10 phần tử kiểu số thực

float A[10];



# Cấp phát bộ nhớ cho mảng



- Các phần tử trong mảng được cấp phát các ô nhớ kế tiếp nhau trong bộ nhớ
- Kích thước của mảng bằng kích thước một phần tử nhân với số phần tử

Ví dụ:

`int A[10]; //Mảng A gồm 10 phần tử nguyên`



Kích thước của mảng A:  $10 \times 4 = 40$  bytes



# Truy nhập phần tử của mảng



- Biến mảng lưu trữ địa chỉ ô nhớ đầu tiên trong vùng nhớ được cấp phát
- Ngôn ngữ C đánh chỉ số các phần tử trong mảng bắt đầu từ 0
- Các phần tử của mảng được truy nhập thông qua:
  - Tên mảng và
  - Chỉ số của phần tử trong mảng

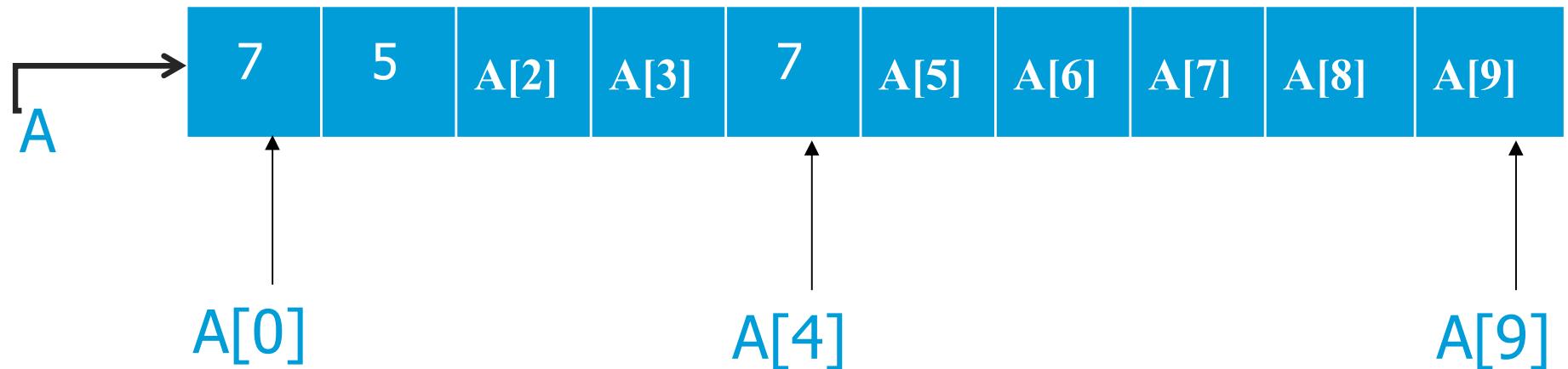
Tên\_Mang[Chỉ\_số\_phần\_tử]



# Ví dụ



```
int A[10]; //Mảng A gồm 10 phần tử nguyên
```



$$A[0] = 7;$$

$$A[1] = 5;$$

$$A[4] = 7;$$

$$\text{int } N = A[1] + A[4]; // N = 12$$



# Ví dụ



```
int A[10];  
for(i = 0; i < 10; i++) A[i]= 2* i;
```

0	2	4	6	8	10	12	14	16	18
---	---	---	---	---	----	----	----	----	----

i : 10

**Chú ý:** C không kiểm tra vượt quá giới hạn của mảng khi truy nhập

```
int A[3], B[4], C[3];
```

A[0]	A[1]	A[2]	B[0]	B[1]	B[2]	B[3]	C[0]	C[1]	C[2]
------	------	------	------	------	------	------	------	------	------

A[5] ⇔ B[2] ⇔ C[-2] ←nếu cấp phát liên tiếp



# Mảng nhiều chiều

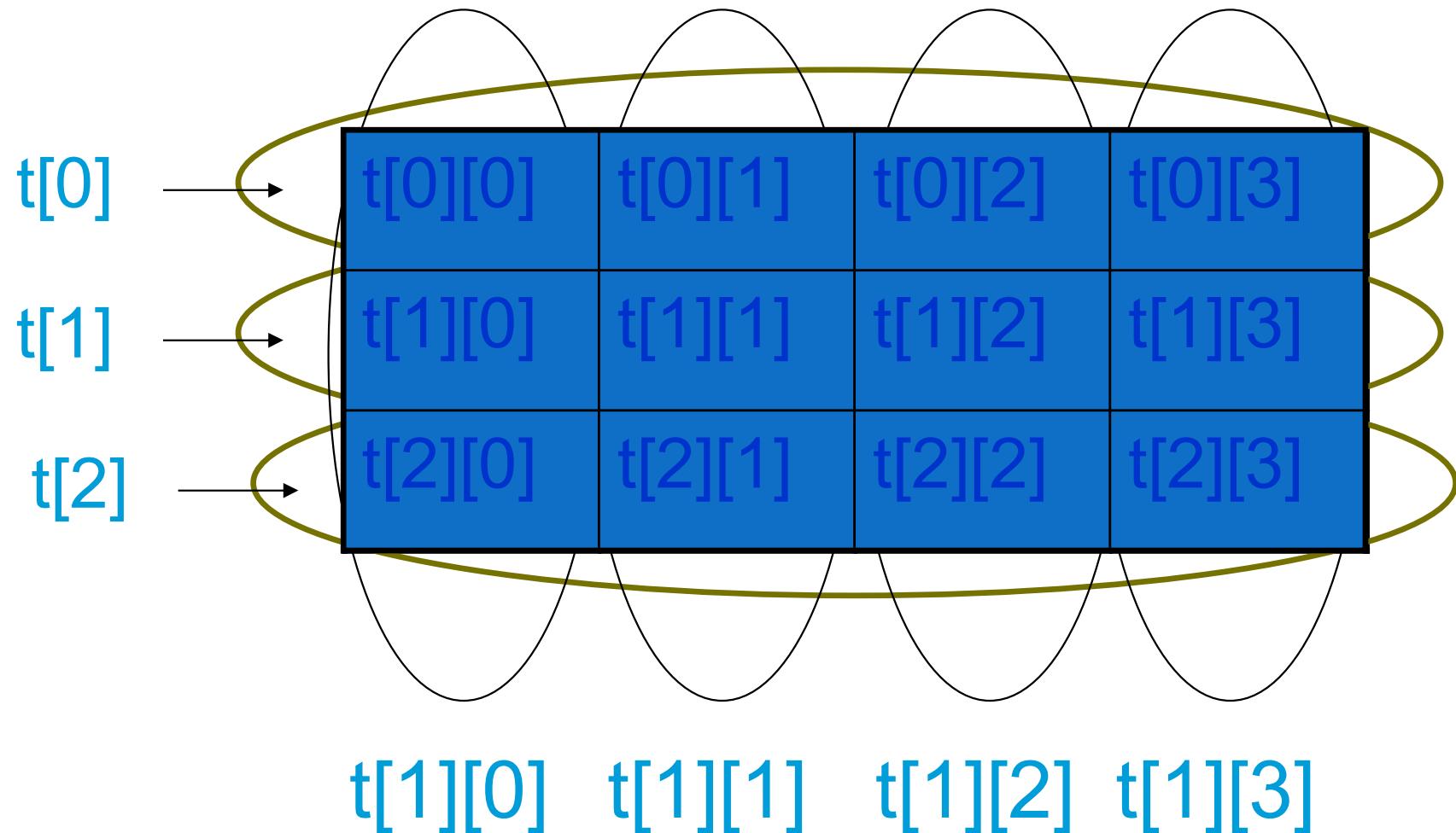


- Mỗi phần tử của mảng có thể là một mảng: Mảng nhiều chiều

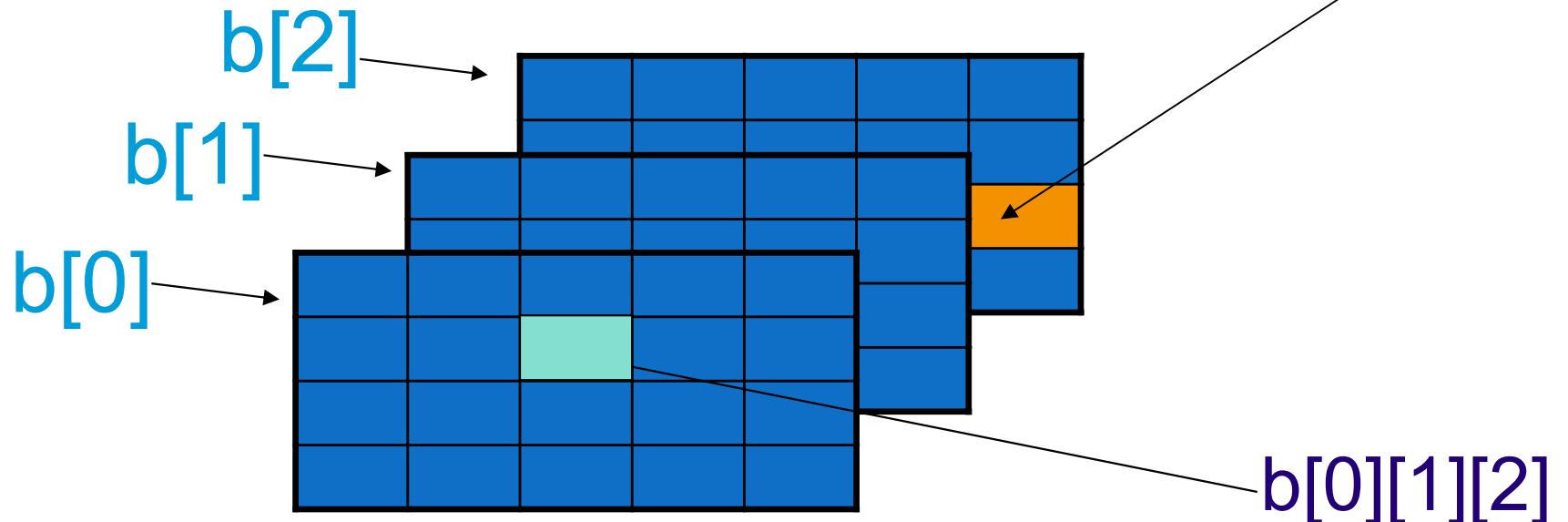
Kiểu\_dữ\_liệu **Tên\_mảng**[Chiều\_1] [Chiều\_2]... [Chiều\_N];

- **Kiểu\_dữ\_liệu:** Kiểu của mỗi phần tử trong mảng
- **Chiều\_1, Chiều\_2,..., Chiều\_N:** Các hằng số nguyên, cho biết kích thước (số phần tử) của mỗi chiều.
- **Mảng gồm:** **Chiều\_1 x Chiều\_2 x...x Chiều\_N** phần tử được lưu trữ trong vùng nhớ liên tục. Các phần tử thuộc kiểu **Kiểu\_dữ\_liệu**.

```
int t[3][4];
```



```
int b[3][4][5];
```



- Mảng  $b$  gồm 3 phần tử  $b[0]$ ,  $b[1]$ ,  $b[2]$
- Mỗi phần tử là mảng hai chiều gồm 4 hàng (hàng 0, 1, 2, 3) và 5 cột (0, 1, 2, 3, 4)
- Mỗi phần tử là một số nguyên có dấu 4 byte



# Khởi tạo giá trị cho mảng



Các phần tử của mảng có thể được khởi tạo giá trị ngay khi khai báo

Ví dụ:

```
int a[4] = {1,4,6,2};
```

```
int b[2][3]={{{1,2,3},{4,5,6}}};
```

```
int t[3][4] = {{1, 2, 3, 4},  
                {5, 6, 7, 8},  
                {9, 10, 11, 12},  
                };
```



# Khởi tạo giá trị cho mảng → Chú ý



- Số lượng giá trị khởi tạo không được lớn hơn số lượng phần tử trong mảng
  - Nếu số lượng này nhỏ hơn, các phần tử còn lại được khởi tạo giá trị 0

```
int A[3][4] = { {1}, {4,5} };  
int A[3][4] = { };//Tất cả đều mang giá trị 0
```
- Có thể xác định kích thước mảng thông qua số giá trị khởi tạo nếu để rõ kích thước mảng

```
int A[8] = {2, 4, 6, 8, 10, 12, 14, 16};  
int B[] = {2, 4, 6, 8, 10, 12, 14, 16};
```



# Các thao tác thường gặp



- Nhập/Xuất dữ liệu cho mảng
  - Mảng 1 chiều, ma trận
- Bài toán đếm
  - Đếm số phần tử
  - Tính toán trên các phần tử..
- Tìm kiếm phần tử
  - Lớn nhất/nhỏ nhất/bất kỳ
- Sắp xếp phần tử trong mảng
  - Theo thứ tự, theo nguyên tắc
- Chèn thêm phần tử, xóa phần tử



# Nhập dữ liệu: Dùng hàm scanf()



## Ví dụ:

```
int T[10];
```

- Nhập dữ liệu cho một phần tử

```
scanf("%d", &T[2]); //phần tử thứ 3 của mảng
```

- Nhập dữ liệu cho cả mảng

- Dùng vòng lặp for

```
for(i = 0; i<10; i++)
```

```
    scanf("%d", &T[i]);
```

- Nên in ra chỉ số phần tử khi nhập

```
printf("T[%d]: ", i);
```

```
scanf("%d", &T[i])
```



# Nhập dữ liệu → Ví dụ 1



## Nhập vào lượng mưa (mm) trong năm

```
#include <stdio.h>
#define MONTHS 12
int main(){
    int rainfall[MONTHS], i;
    for(i=0; i < MONTHS; i++)
    {
        printf("Nhập lượng mưa tháng %d: ", i+1);
        scanf("%d", &rainfall[i]);
    }
    return 0;
}
```



# Nhập dữ liệu → Lưu ý



- Nếu số phần tử của mảng chỉ được biết tại thời điểm thực hiện chương trình (nhưng *biết số phần tử tối đa*)
  - Khai báo mảng với kích thước tối đa
  - Sử dụng biến nguyên lưu số phần tử thực sự của mảng.

## Ví dụ:

- Nhập vào mảng không quá 100 số thực
  - Khai báo mảng thực có tối đa 100 phần tử.
  - Nhập số phần tử thực sự của mảng
  - Nhập giá trị cho từng phần tử (dùng for)



# Nhập dữ liệu → Ví dụ 2



```
#include<stdio.h>
int main(){
    float A[100];
    int n, i;
    do{
        printf("Cho biet so phan tu cua mang: ");
        scanf("%d",&n);
    }while (n>100||n<=0);
    for(i = 0; i < n; i++){
        printf("A[%d] = ", i); scanf("%f", &A[i]);
    }
    return 0;
}
```



# Nhập dữ liệu → Ví dụ 2



```
#include<stdio.h>
int main(){
    double A[100];
    int n, i;
    do{
        printf("Cho biet so phan tu cua mang: ");
        scanf("%d",&n);
    }while (n>100||n<=0);
    for(i = 0; i < n; i++){
        printf("A[%d] = ", i); scanf("%lf",&A[i]);
    }
    return 0;
}
```



# Xuất dữ liệu từ mảng: hàm printf()



## Ví dụ:

```
int T[10];
```

- Hiển thị phần tử thứ 5:

```
printf("%d", T[4]);
```

- Để hiển thị tất cả các phần tử:

```
for(i = 0; i < 10; i++)
```

```
    printf("%4d", T[i]);
```

## Các kiểu xuất dữ liệu

- Hiển thị tất cả/một phần theo dòng/cột..
- Hiển thị từng k phần tử trên một dòng...



# Xuất dữ liệu trong mảng → Ví dụ 1



```
#include <stdio.h>
#define MAX 12
int main(){
    int A[MAX], i;
    for ( i=0; i < MAX; i++ ){
        printf("A[%d]: ", i+1);
        scanf("%d", &A[i]);
    }
    printf("\n");
    for(i=0;i < MAX; i++)
        printf("%4d", A[i]);
    printf("\n");
    for(i=0; i < MAX; i++)
        printf("%d\n", A[i]);
    printf("\n");
    for(i=0; i < MAX; i++){
        printf("%4d", A[i]);
        if( (i+1) %4==0) printf("\n");
    }
    return 0;
}
```

1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12



## Ví dụ 2: Nhập và đưa ra màn hình một ma trận



```
Nhap so hang: 3
```

```
Nhap so cot: 3
```

```
Nhap phan tu A[1,1]: 1
```

```
Nhap phan tu A[1,2]: 2
```

```
Nhap phan tu A[1,3]: 3
```

```
Nhap phan tu A[2,1]: 4
```

```
Nhap phan tu A[2,2]: 5
```

```
Nhap phan tu A[2,3]: 6
```

```
Nhap phan tu A[3,1]: 7
```

```
Nhap phan tu A[3,2]: 8
```

```
Nhap phan tu A[3,3]: 9
```

```
MA TRAN DA NHAP
```

1	2	3
4	5	6
7	8	9

```
-----  
Process exited after 10.68 second  
Press any key to continue . . .
```

```
Nhap so hang: 2
```

```
Nhap so cot: 3
```

```
Nhap phan tu A[1,1]: 1
```

```
Nhap phan tu A[1,2]: 2
```

```
Nhap phan tu A[1,3]: 3
```

```
Nhap phan tu A[2,1]: 4
```

```
Nhap phan tu A[2,2]: 5
```

```
Nhap phan tu A[2,3]: 6
```

```
MA TRAN DA NHAP
```

1	2	3
4	5	6

```
-----  
Process exited after 5.943 second  
Press any key to continue . . .
```



# Ví dụ 2 → Kết quả thực hiện



```
#include <stdio.h>
int main(){
    int A[20][20], n, m, i,j;
    printf("Nhập số hàng: "); scanf("%d",&n);
    printf("Nhập số cột: "); scanf("%d",&m);
    printf("\n");
    for(i=0;i < n;i++)
        for(j=0;j < m;j++){
            printf("Nhập phần tử A[%d,%d]: ", i+1,j+1);
            scanf("%d", &A[i][j]);
        }
    printf("\n\nMÃ TRAN DA NHẬP\n\n");
    for(i=0;i < n; i++){
        for(j=0;j < m; j++)
            printf("%4d", A[i][j]);
        printf("\n");
    }
    return 0;
}
```

```
Nhập số hàng: 2
Nhập số cột: 3
Nhập phần tử A[1,1]: 1
Nhập phần tử A[1,2]: 2
Nhập phần tử A[1,3]: 3
Nhập phần tử A[2,1]: 4
Nhập phần tử A[2,2]: 5
Nhập phần tử A[2,3]: 6
MA TRAN DA NHẬP
      1   2   3
      4   5   6
```



# Đếm số phần tử thỏa mãn điều kiện



- Duyệt từng phần tử của dãy (có thể dùng vòng lặp **for**)
- Nếu phần tử xét thỏa mãn điều kiện
  - Ghi nhận
- Chuyển sang xem xét phần tử tiếp theo

**Ví dụ:** Đếm số tháng có lượng mưa lớn hơn 50mm

```
int dem = 0;  
for(i = 0; i < MONTHS; i++)  
    if(rainfall[i] > 50)  
        dem++;  
printf("\nSo thang mua nhieu hon 50mm: %d", dem);
```



## Ví dụ: Nhập mảng n số nguyên, đưa ra trung bình cộng các số chia hết cho 7 trong mảng



Số phần tử của mảng ( $n < 100$ )  $n = 6$

$A[0] = 12$

$A[1] = 14$

$A[2] = 7$

$A[3] = 8$

$A[4] = 6$

$A[5] = 21$

TBC số chia hết cho 7: 14.00.

-----

Process exited after 15.05 seconds

Press any key to continue . . .



# Ví dụ: Nhập mảng n số nguyên, đưa ra trung bình cộng các số chia hết cho 7 trong mảng



```
#include<stdio.h>
int main(){
    int A[100], n, i, d = 0, S = 0;
    printf("So phan tu cua mang (n<100) n = "); scanf("%d",&n);
    for(i = 0; i < n; i++){
        printf("A[%d] = ", i);
        scanf("%d",&A[i]);
    }
    for(i = 0; i < n; i++)
        if(A[i] %7==0){
            d++;
            S+= A[i];
        }
    if(d > 0)
        printf("TBC so chia het cho 7: %7.2f.",(float)S/d);
    else
        printf("Trong day khong co so chia het cho 7.");
    return 0;
}
```

```
So phan tu cua mang (n<100) n = 5
A[0] = 14
A[1] = 7
A[2] = 23
A[3] = 28
A[4] = 45
TBC so chia het cho 7: 16.33.
Process exited after 16.44 seconds
Press any key to continue . . .
```



# Tìm kiếm phần tử



## Tìm phần tử lớn nhất (*nhỏ nhất*)

- Giả sử phần tử đó là phần tử đầu tiên
- Lần lượt so sánh với các phần tử còn lại
  - Nếu phần tử mới của dãy lớn hơn  $\Rightarrow$  coi đây là phần tử lớn nhất và tiếp tục so sánh với phần tử kế tiếp
  - Nếu không đúng, so sánh tiếp với phần tử kế tiếp

### Ví dụ:

Tìm tháng có lượng mưa nhiều nhất trong năm?

```
max = rainfall[0];
for(i = 1; i < MONTHS; i++)
    if(rainfall[i] > max)
        max = rainfall[i];
printf("\n Luong mua nhieu nhat la: %d", max);
```



# Tìm kiếm phần tử



- ❖ Tìm kiếm các phần tử thỏa mãn điều kiện (*giống bài toán đếm*)
  - Dùng for duyệt toàn bộ
  - Nếu cần thiết, dùng thêm mảng ghi lại chỉ số

## Ví dụ:

Đưa ra danh sách các tháng có lượng mưa nhiều hơn 50mm.

```
printf("Thang co luong mua lon hon 50mm");
for(i = 0; i < MONTHS; i++)
    if(rainfall[i] > 50)
        printf("\nThang %d", i+1);
```



# Tìm kiếm phần tử (tiếp)



- Tìm phần tử đầu tiên của danh sách
  - Dùng vòng lặp for kết hợp với break;
  - Dùng vòng lặp while

**Ví dụ:** Đưa ra phần tử đầu tiên của mảng có giá trị bằng k cho trước.



# Tìm kiếm phần tử → Ví dụ



```
int Table[100];
```

```
int N, i, k, f; //N: số phần tử, k: phần tử cần tìm
```

Dùng vòng lặp for:

```
for(i = 0; i < N; i++)
    if(Table[i] == k) break;
if(i< N) printf("Tim thay tai vi tri %d", i);
```

Dùng vòng lặp while:

```
i=0; f =0; //f: found, f = 1 ⇔ k is found
```

```
while(i < N && f==0){
```

```
    if(Table[i] == k)
```

```
        f = 1;
```

```
    else i++;
```

```
}
```

```
if (f==1) printf("Tim thay tai vi tri %d", i);
```



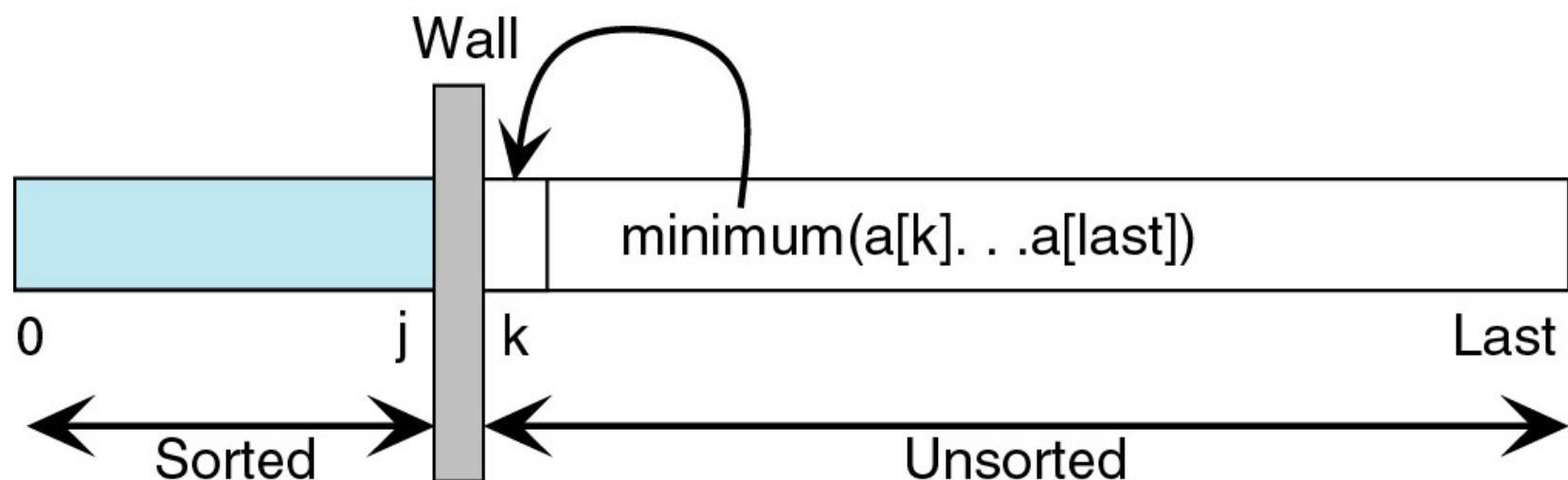
# Bài toán sắp xếp theo thứ tự



- Cho mảng phần tử, sắp xếp theo thứ tự tăng/giảm
- Các thuật toán
  - Sắp xếp thêm dần (Insertion sort)
  - Sắp xếp lựa chọn (Selection sort)
  - Sắp xếp nổi bọt (Bubble sort)
  - Sắp xếp vun đống (Heap sort)
  - Sắp xếp nhanh (Quick sort)
  - Sắp xếp trộn (Merge sort)

Nguyên tắc: Tại lượt sắp thứ k, tìm phần tử nhỏ nhất trong số các phần tử chưa được sắp xếp ( $[k..last]$ ) và đổi chỗ cho phần tử thứ k (có chỉ số  $k-1$ )

- Khi  $k = 1$ , phần tử thứ nhất (chỉ số 0) đúng vị trí
- Khi  $k = 2$ , phần tử thứ hai (chỉ số 1) đúng vị trí...



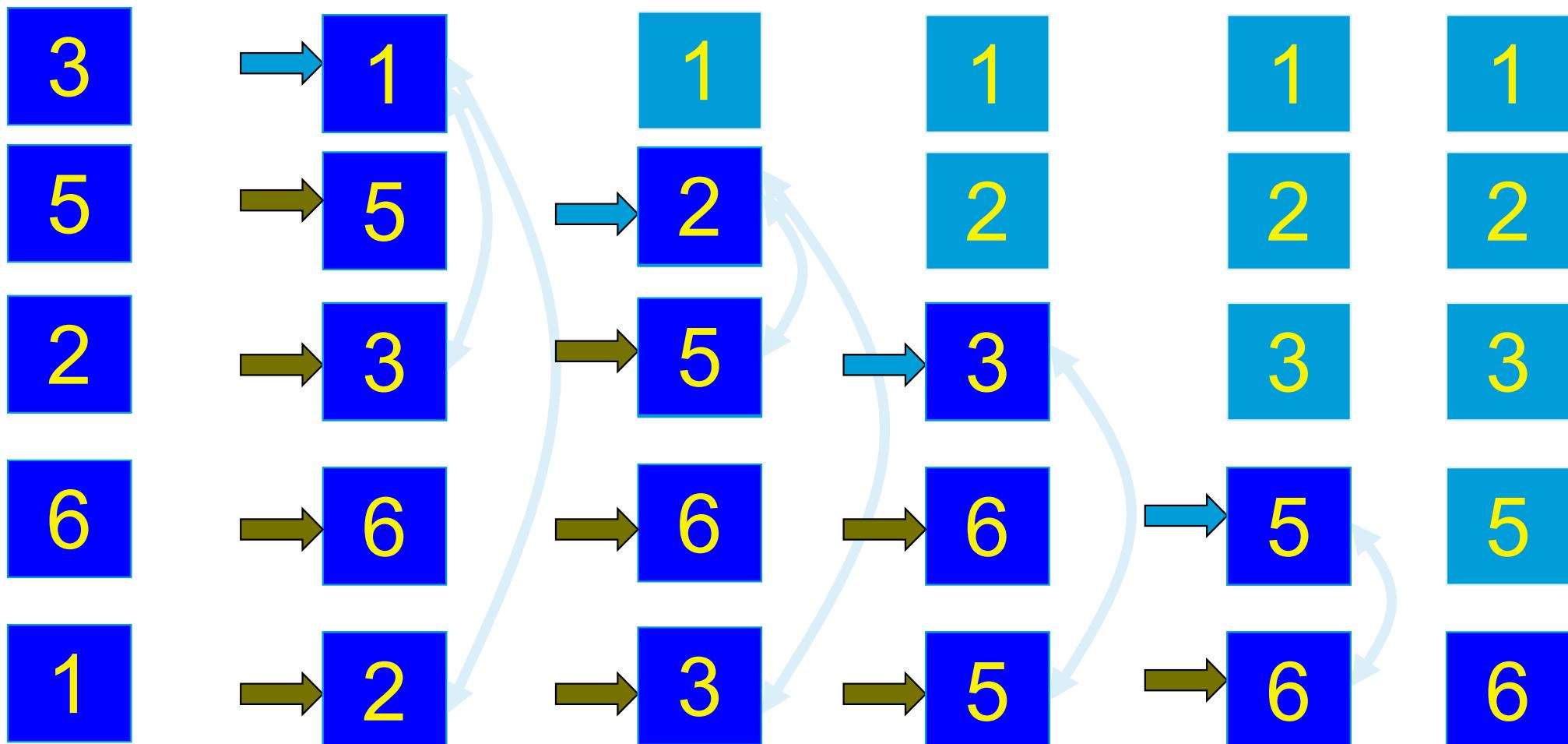
Dãy

Lượt 1

Lượt 2

Lượt 3

Lượt 4





# Bài toán sắp xếp tăng → Thuật toán lựa chọn



```
//Khai báo các biến
int A[100];      //Mảng chứa dữ liệu
int N, i, j, tmp;
//Sắp xếp
for(i = 0; i < N - 1; i++)
    for(j = i + 1; j < N; j++)
        if(A[i] > A[j]) {
            tmp = A[i];
            A[i] = A[j];
            A[j] = tmp;
        }
```



# Ví dụ



- Nhập vào từ bàn phím một mảng các số nguyên không quá 100 phần tử
- Hiển thị dãy số vừa nhập
- Sắp xếp dãy theo thứ tự giảm dần
- Hiển thị dãy tại mỗi lượt sắp xếp

Số phần tử [NK100], N = 5

Hay nhập dãy số...

```
A[1] = 1  
A[2] = 2  
A[3] = 3  
A[4] = 4  
A[5] = 5
```

Dãy vừa nhập...

1	2	3	4	5	
Sắp xếp dãy theo thuật toán lục chọn:					
Luot 1 :	5	1	2	3	4
Luot 2 :	5	4	1	2	3
Luot 3 :	5	4	3	1	2
Luot 4 :	5	4	3	2	1

Process exited after 4.969 seconds with  
Press any key to continue . . .



# Ví dụ



```
int main(){
    int A[100], N, i, j, t;
    printf("So phan tu [N<100], N = "); scanf("%d",&N);
    printf("Hay nhap day so...\\n");
    for(i=0; i < N; i++){
        printf("A[%d] = ",i+1); scanf("%d",&A[i]);
    }
    printf("\\nDay vua nhap...\\n");
    for(i=0; i < N; i++)
        printf("%4d", A[i]);
    printf("\\nSap xep day theo thuat toan lua chon:");
    for(i=0; i < N-1; i++){
        for(j=i+1; j < N; j++)
            if(A[i] < A[j]){
                t = A[i];
                A[i] = A[j];
                A[j] = t;
            }
        printf("\\nLuot %d : ",i+1);
        for(j=0;j < N; j++)
            printf("%4d", A[j]);
    }
}
```

```
So phan tu [N<100], N = 5
Hay nhap day so...
A[1] = 1
A[2] = 2
A[3] = 3
A[4] = 4
A[5] = 5

Day vua nhap...
      1   2   3   4   5
Sap xep day theo thuat toan lua chon:
Luot 1 :      5   1   2   3   4
Luot 2 :      5   4   1   2   3
Luot 3 :      5   4   3   1   2
Luot 4 :      5   4   3   2   1
-----
Process exited after 4.969 seconds with
Press any key to continue . . .
```



# Ví dụ



```
So phan tu [NK100], N = -12
```

```
So phan tu [NK100], N = 120
```

```
So phan tu [NK100], N = 5
```

```
Hay nhap day so...
```

```
A[1] = 1
```

```
A[2] = 4
```

```
A[3] = 5
```

```
A[4] = 7
```

```
A[5] = 2
```

```
Day vua nhap...
```

```
1 4 5 7 2
```

```
Sap xep day theo thuat toan lua chon:
```

```
Luot 1 : 7 1 4 5 2
```

```
Luot 2 : 7 5 1 4 2
```

```
Luot 3 : 7 5 4 1 2
```

```
Luot 4 : 7 5 4 2 1
```

```
-----
```

```
Process exited after 13.97 seconds with return value 4
```

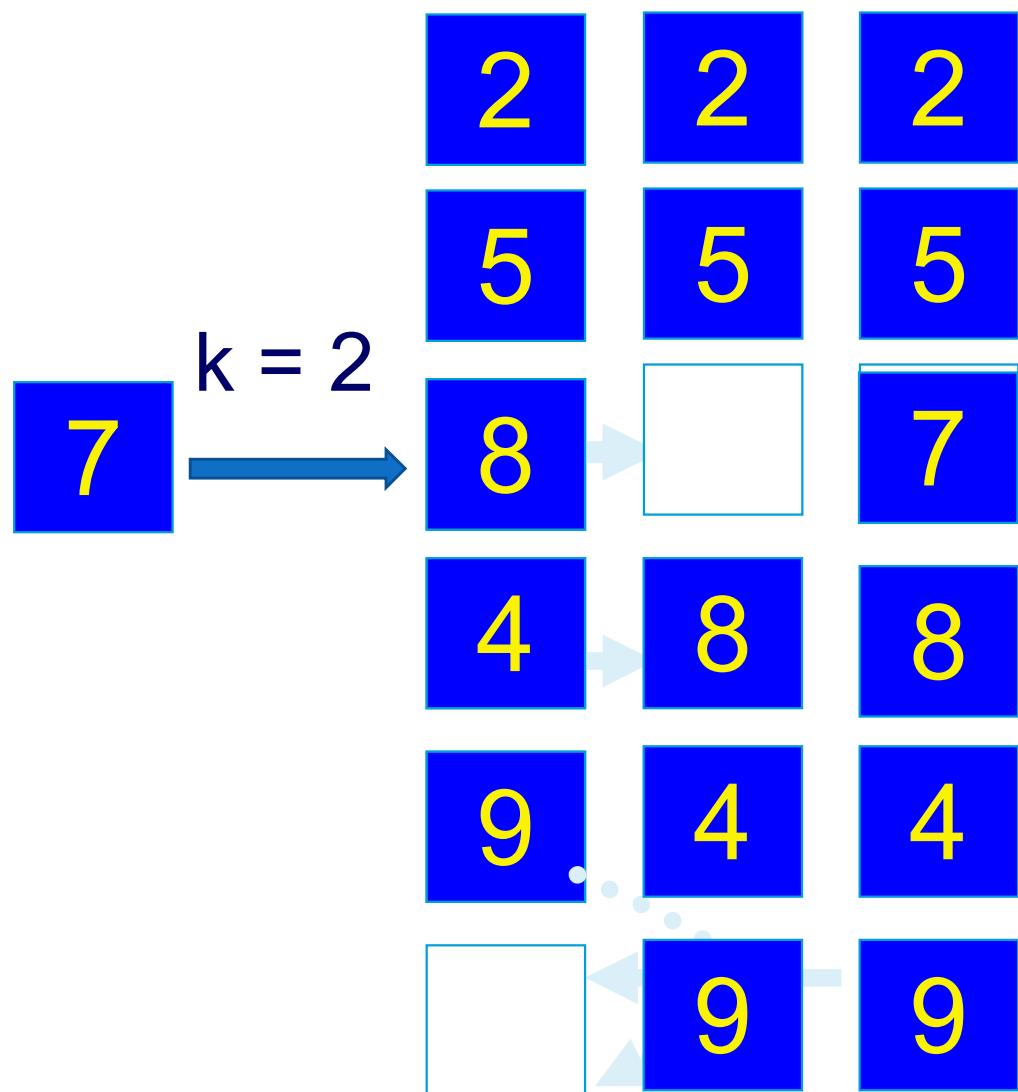
```
Press any key to continue . . .
```



# Ví dụ



```
int A[100], N, i, j, t;
do{
    printf("So phan tu [N<100], N = "); scanf("%d",&N);
}while(N<=0 || N>100);
printf("Hay nhap day so...\\n");
for(i=0; i < N; i++){
    printf("A[%d] = ",i+1); scanf("%d",&A[i]);
}
printf("\\nDay vua nhap...\\n");
for(i=0; i < N; i++)
    printf("%4d", A[i]);
printf("\\nSap xep day theo thuat toan lua chon:");
for(i=0; i < N-1; i++){
    for(j=i+1; j < N; j++)
        if(A[i] < A[j]){
            t = A[i];
            A[i] = A[j];
            A[j] = t;
        }
    printf("\\nLuot %d : ",i+1);
    for(j=0;j < N; j++)
        printf("%4d", A[j]);
}
```



```
for(i = N; i > k; i--)
```

```
    A[i] = A[i-1];
```

```
A[k] = x;
```

```
N = N + 1;
```

Chú ý:

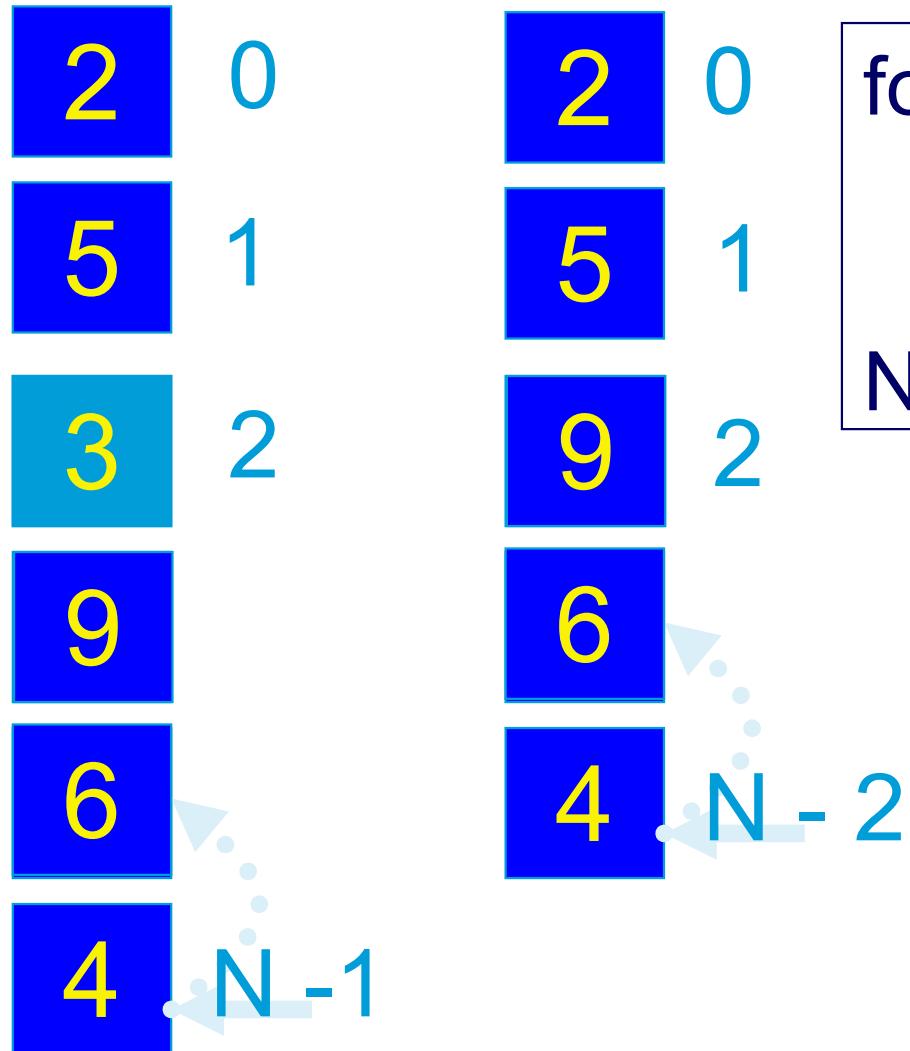
$N = \text{MAX}$ : không chèn được

$k \geq N$ : Chèn vào vị trí N

$k < 0$ : Chèn vào vị trí 0



# Bài toán xóa phần tử ở vị trí k ( $0 \leq k < N$ )



```
for(i = k+1; i < N; i++)  
    A[i-1] = A[i];  
    N = N - 1;
```

N phần tử

N-1 phần tử



# Bài tập



1. Nhập vào từ bàn phím một dãy số nguyên ( $<100$  phần tử). Sắp xếp dãy theo nguyên tắc: Bên trên là số chẵn chia hết cho 3. Bên dưới là số lẻ chia hết cho 3. Giữa là các số còn lại. Đưa cả 2 dãy ra màn hình.
2. Đọc vào dãy số có  $n$  phần tử ( $n < 100$ ). Đọc số  $x$  và số  $k$  nguyên. Chèn  $x$  vào vị trí  $k$  của dãy. Nếu  $k \geq n$ , chèn  $x$  vào vị trí  $n$ .
3. Nhập vào một dãy số ( $<100$  phần tử) và sắp xếp theo thứ tự tăng dần. Nhập thêm vào một số và chèn số mới nhập vào đúng vị trí.
4. Nhập vào một dãy ( $<100$  phần tử); xóa đi các phần tử chia hết cho 5 và đưa kết quả ra màn hình



# Bài tập



Nhập vào từ bàn phím một dãy số nguyên (<100 phần tử). Sắp xếp dãy theo nguyên tắc: Bên trên là số chẵn chia hết cho 3. Bên dưới là số lẻ chia hết cho 3. Giữa là các số còn lại. Đưa cả 2 dãy ra màn hình.

```
So phan tu N = 10
A[0] = 1
A[1] = 2
A[2] = 3
A[3] = 4
A[4] = 5
A[5] = 6
A[6] = 9
A[7] = 12
A[8] = 15
A[9] = 4

In day da nhap:
    1   2   3   4   5   6   9   12  15   4
In lai day da sap theo yeu cau:
    6   12  4   5   1   2   4   3   15  9
-----
```



# Minh họa bài 1



```
1 #include<stdio.h>
2 int main(){
3     int A[100], N, i;
4     do{
5         printf("So phan tu N = ");
6         scanf("%d",&N);
7     }while(N<0 || N >=100);
8
9     for(i = 0; i < N; i++){
10        printf("A[%d] = ",i);
11        scanf("%d", &A[i]);
12    }
13    printf("\nIn day da nhap:\n");
14    for(i = 0; i < N; i++)
15        printf("%4d", A[i]);
16    //Sap xep so chan chia het cho 3 len truoc
17    int d = 0, t;
18    for(i = 0;i < N; i++)
19        if(A[i]%6==0){
20            t = A[i];
21            A[i] = A[d];
22            A[d] = t;
23            d++;
24        }
```

```
25 //Sap xep so le chia het cho 3 o cuoi day
26 for(i = d; i < N; i++)
27     if(A[i]%3 != 0){
28         t = A[i];
29         A[i] = A[d];
30         A[d] = t;
31         d++;
32     }
33 printf("\nIn lai day da sap theo yeu cau:\n");
34 for(i = 0; i < N; i++)
35     printf("%4d", A[i]);
36
37 }
```

```
So phan tu N = 10
A[0] = 1
A[1] = 2
A[2] = 3
A[3] = 4
A[4] = 5
A[5] = 6
A[6] = 9
A[7] = 12
A[8] = 15
A[9] = 4
```

```
In day da nhap:
1 2 3 4 5 6 9 12 15 4
In lai day da sap theo yeu cau:
6 12 4 5 1 2 4 3 15 9
```



## Bài tập 2



Đọc vào dãy số có n phần tử ( $n < 100$ ). Đọc số x và số k nguyên.  
Chèn x vào vị trí k của dãy.

- Nếu  $k < 0$  thì chèn x vào vị trí thứ 0.
- Nếu  $k \geq n$ , chèn x vào vị trí n.
- Nếu  $0 \leq k < n$  thì chèn x vào vị trí k.

```
So phan tu N = 5
A[0] = 1
A[1] = 2
A[2] = 3
A[3] = 4
A[4] = 5
In day da nhap:
 1 2 3 4 5
Nhap so x = 0
Nhap so nguyen k = -1
In lai day sau khi chen x = 0:
 0 1 2 3 4 5
```

```
So phan tu N = 6
A[0] = 1
A[1] = 2
A[2] = 3
A[3] = 4
A[4] = 5
A[5] = 6
In day da nhap:
 1 2 3 4 5 6
Nhap so x = 0
Nhap so nguyen k = 2
In lai day sau khi chen x = 0:
 1 2 0 3 4 5 6
```

```
So phan tu N = 5
A[0] = 5
A[1] = 4
A[2] = 3
A[3] = 2
A[4] = 1
In day da nhap:
 5 4 3 2 1
Nhap so x = -2
Nhap so nguyen k = 6
In lai day sau khi chen x = -2:
 5 4 3 2 1 -2
```



# Minh họa bài tập 2



```
1 #include<stdio.h>
2 #define MAX 100
3 int main(){
4     int A[MAX], N, i;
5     do{
6         printf("So phan tu N = ");
7         scanf("%d",&N);
8     }while(N<0 || N >=100);
9
10    for(i = 0; i < N; i++){
11        printf("A[%d] = ",i);
12        scanf("%d", &A[i]);
13    }
14    printf("In day da nhap:\n");
15    for(i = 0; i < N; i++)
16        printf("%4d", A[i]);
17    int x, k;
18    printf("\nNhập số x = "); scanf("%d",&x);
19    printf("Nhập số nguyên k = "); scanf("%d",&k);
20    if (N == MAX)
21        printf("Mang day. Khong chen them duoc!");
```

```
22    else{
23        if (k<0) k = 0;
24        if (k>=0 && k<N)
25        {
26            for(i = N; i > k; i--)
27                A[i] = A[i-1];
28            A[k] = x;
29            N++;
30        }
31        else if(k>=N)
32        {
33            A[N] = x;
34            N++;
35        }
36        printf("In lai day sau khi chen x = %d:\n", x);
37        for(i = 0; i < N; i++)
38            printf("%4d", A[i]);
39    }
40    return 0;
41 }
```

```
So phan tu N = 5
A[0] = 1
A[1] = 2
A[2] = 3
A[3] = 4
A[4] = 5
In day da nhap:
 1 2 3 4 5
Nhập số x = 0
Nhập số nguyên k = -1
In lai day sau khi chen x = 0:
 0 1 2 3 4 5
```

```
So phan tu N = 6
A[0] = 1
A[1] = 2
A[2] = 3
A[3] = 4
A[4] = 5
A[5] = 6
In day da nhap:
 1 2 3 4 5 6
Nhập số x = 0
Nhập số nguyên k = 2
In lai day sau khi chen x = 0:
 1 2 0 3 4 5 6
```

```
So phan tu N = 5
A[0] = 5
A[1] = 4
A[2] = 3
A[3] = 2
A[4] = 1
In day da nhap:
 5 4 3 2 1
Nhập số x = -2
Nhập số nguyên k = 6
In lai day sau khi chen x = -2:
 5 4 3 2 1 -2
```



# Bài tập 3



Nhập vào một dãy số (<100 phần tử) và sắp xếp theo thứ tự tăng dần. Nhập thêm vào một số x và chèn số mới nhập vào đúng vị trí.

```
So phan tu N = 6
A[0] = 1
A[1] = 6
A[2] = 5
A[3] = 7
A[4] = 8
A[5] = 9
In day da nhap:
    1   6   5   7   8   9
In lai day sau khi xep tang dan:
    1   5   6   7   8   9
Nhap so x = 4
In lai day sau khi chen x = 4:
    1   4   5   6   7   8   9
-----
```



# Bài 3: Sắp xếp tăng dần và chèn đúng vị trí



```
1 #include<stdio.h>
2 #define MAX 100
3 int main(){
4     int A[MAX], N, i, j, t;
5     do{
6         printf("So phan tu N = ");
7         scanf("%d",&N);
8     }while(N<0 || N >=100);
9
10    for(i = 0; i < N; i++){
11        printf("A[%d] = ",i);
12        scanf("%d", &A[i]);
13    }
14    printf("Day da nhap:\n");
15    for(i = 0; i < N; i++)
16        printf("%4d", A[i]);
17
18    for(i = 0; i < N-1; i++)
19        for(j = i; j < N; j++)
20            if (A[i]>A[j]){
21                t = A[i];
22                A[i] = A[j];
23                A[j] = t;
24            }
25    printf("\nDay xep tang dan:\n");
26    for(i = 0; i < N; i++)
27        printf("%4d", A[i]);
```

```
29    int x;
30    printf("\nNhập số x = "); scanf("%d",&x);
31    if (N == MAX)
32        printf("Mang day!");
33    else{
34        i = N;
35        while((i > 0) &&(A[i-1] > x)) {
36            A[i] = A[i-1];
37            i--;
38        }
39        A[i] = x;
40        N++;
41        printf("Day sau khi chen x = %d:\n", x);
42        for(i = 0; i < N; i++)
43            printf("%4d", A[i]);
44    }
45    return 0;
46 }
```

```
So phan tu N = 6
A[0] = 1
A[1] = 6
A[2] = 5
A[3] = 7
A[4] = 8
A[5] = 9
In day da nhap:
1 6 5 7 8 9
In lai day sau khi xep tang dan:
1 5 6 7 8 9
Nhập số x = 4
In lai day sau khi chen x = 4:
1 4 5 6 7 8 9
```



## Bài tập 4

Nhập vào một dãy gồm n phần tử ( $0 < n < 100$ ). Xóa đi các phần tử chia hết cho 5 và đưa kết quả ra màn hình.

```
So phan tu N = 101
So phan tu N = 6
A[0] = 5
A[1] = 1
A[2] = 3
A[3] = 10
A[4] = 25
A[5] = 3
Day da nhap:
      5   1   3   10  25   3
Day sau khi xoa:
      1   3   3
-----
```



# Minh họa bài 4: Xóa phần tử chia hết cho 5



```
#include<stdio.h>
#define MAX 100
int main(){
    int A[MAX], N, i;
    do{
        printf("So phan tu N = ");
        scanf("%d",&N);
    }while(N<=0 || N >=100);

    for(i = 0; i < N; i++){
        printf("A[%d] = ",i);
        scanf("%d", &A[i]);
    }
    printf("Day da nhap:\n");
    for(i = 0; i < N; i++)
        printf("%4d", A[i]);
}
```

```
// giu lai cac phan tu khong chia het cho 5
{
    int d = 0;
    for(i = 0;i < N; i++)
        if(A[i] % 5 != 0){
            A[d] = A[i];
            d++;
        }
    N = d;// cap nhat so phan tu sau xoa
}
printf("\nDay sau khi xoa:\n");
for(i = 0; i < N; i++)
    printf("%4d", A[i]);
return 0;
}
```



# Bài tập về Ma trận



1. Viết chương trình nhập vào một ma trận vuông, các phần tử nguyên, sau đó
  - Đưa ra ma trận tam giác dưới
  - Đưa ra ma trận tam giác trên
2. Nhập M, N ( $M, N < 30$ ) và một ma trận  $M \times N$ . Đưa ma trận ra màn hình
  - Tìm hàng/cột có tổng các phần tử lớn nhất
  - Tìm số lớn nhất/nhỏ nhất và vị trí trong ma trận
  - Đưa ra ma trận S cùng kích thước thỏa mãn

$$S_{ij} = \begin{cases} 1 & \text{nếu } A_{ij} > 0 \\ 0 & \text{nếu } A_{ij} = 0 \\ -1 & \text{nếu } A_{ij} < 0 \end{cases}$$



# Bài tập về Ma trận



Viết chương trình nhập vào một ma trận vuông, các phần tử nguyên, sau đó

- Đưa ra ma trận tam giác dưới
- Đưa ra ma trận tam giác trên

Nhap kích thuoc: 3

```
Nhap phan tu [1,1] = 1
Nhap phan tu [1,2] = 2
Nhap phan tu [1,3] = 3
Nhap phan tu [2,1] = 4
Nhap phan tu [2,2] = 5
Nhap phan tu [2,3] = 6
Nhap phan tu [3,1] = 7
Nhap phan tu [3,2] = 6
Nhap phan tu [3,3] = 5
```

MA TRAN DA NHAP

1	2	3
4	5	6
7	6	5

MA TRAN TAM GIAC TREN

1	2	3
	5	6
		5

MA TRAN TAM GIAC DUOI

1		
4	5	
7	6	5



# Nhập vào một ma trận vuông,..



```
1 #include <stdio.h>
2 int main(){
3     int A[20][20], N, i, j;
4     printf("Nhập kích thước: "); scanf("%d",&N);
5     printf("\n");
6     for (i=0; i < N; i++)
7     {
8         for(j=0; j < N; j++)
9             {
10                printf("Nhập phần tử [%d,%d] = ", i+1,j+1);
11                scanf("%d", &A[i][j]);
12            }
13        printf("\nMA TRAN DA NHAP\n");
14        for(i=0;i < N; i++){
15            for(j=0;j < N; j++)
16                printf("%4d",A[i][j]);
17            printf("\n");
18        }
19    }
20 }
```

```
MA TRAN DA NHAP
1   2   3
4   5   6
7   6   5
```

```
MA TRAN TAM GIAC TREN
1   2   3
      5   6
          5

MA TRAN TAM GIAC DUOI
1
4   5
7   6   5
```

```
17 printf("\nMA TRAN TAM GIAC TREN\n");
18 for (i=0;i < N; i++){
19     for(j=0;j < N; j++)
20         if(j >= i)
21             printf("%4d",A[i][j]);
22         else
23             printf("%4c",32);
24     printf("\n");
25 }
26 printf("\n MA TRAN TAM GIAC DUOI\n");
27 for (i=0;i<N;i++){
28     for(j=0;j <= i;j++)
29         printf("%4d",A[i][j]);
30     printf("\n");
31 }
32 return 0;
33 }
```



# Ma trận tổng, chuyển vị



Viết chương trình nhập vào 2 ma trận A và B gồm m hàng và n cột, các phần tử nguyên, sau đó:

- Tính và in ra ma trận tổng  $C = A+B$  ( $C_{ij} = A_{ij}+B_{ij}$ )
- In ra ma trận chuyển vị  $C^T$  của ma trận C ( $C^T_{ij} = C_{ji}$ )

Nhap so hang m = 2  
Nhap so cot n = 3

Nhap ma tran A:

A[0][0] = 1  
A[0][1] = 1  
A[0][2] = 1  
A[1][0] = 1  
A[1][1] = 1  
A[1][2] = 1

Nhap ma tran B:

B[0][0] = 2  
B[0][1] = 1  
B[0][2] = 3  
B[1][0] = 4  
B[1][1] = 5  
B[1][2] = 2

Ma tran C = A + B:

3	2	4
5	6	3

Ma tran chuyen vi cua C:

3	5
2	6
4	3



# Ma trận tổng, chuyển vị



```
1 #include<stdio.h>
2 const MAX = 100;
3 int main(){
4     int A[MAX][MAX], B[MAX][MAX], C[MAX][MAX], m, n, i, j;
5     printf("Nhập số hàng m = "); scanf("%d", &m);
6     printf("Nhập số cột n = "); scanf("%d", &n);
7     puts("Nhập ma trận A:");
8     for(i = 0; i < m; i++)
9     {
10        for(j = 0; j < n; j++)
11        {
12            printf("A[%d][%d] = ", i, j);
13            scanf("%d", &A[i][j]);
14        }
15    puts("Nhập ma trận B:");
16    for(i = 0; i < m; i++)
17    {
18        for(j = 0; j < n; j++)
19        {
20            printf("B[%d][%d] = ", i, j);
21            scanf("%d", &B[i][j]);
22        }
23    }
```

```
24    {
25        puts("Ma trận C = A + B:");
26        for(i = 0; i < m; i++)
27        {
28            for(j = 0; j < n; j++)
29            {
30                C[i][j] = A[i][j] + B[i][j];
31                printf("%d \t", C[i][j]);
32            }
33        }
34        printf("\n");
35    }
36    puts("Ma trận chuyển vị của C:");
37    for(i = 0; i < n; i++)
38    {
39        for(j = 0; j < m; j++)
40        {
41            printf("%d \t", C[j][i]);
42        }
43        printf("\n");
44    }
45}
```



# Ma trận tích $C = AB$



Viết chương trình nhập ma trận A gồm m hàng và k cột, ma trận B gồm k hàng và n cột, các phần tử nguyên, sau đó:

- Tính và in ra ma trận tích  $C = A \cdot B$
- In ra ma trận chuyển vị của ma trận C.

```
Nhap m = 2
Nhap k = 3
Nhap n = 2
Nhap ma tran A:
A[0][0] = 1
A[0][1] = 1
A[0][2] = 2
A[1][0] = 3
A[1][1] = 4
A[1][2] = 5
Nhap ma tran B:
B[0][0] = 3
B[0][1] = 2
B[1][0] = 1
B[1][1] = 4
B[2][0] = 5
B[2][1] = 6
Ma tran C = A*B:
```

```
14 18
38 52
```

```
Ma tran chuyen vi cua C:
14 38
18 52
```



# Ma trận tích C = AB



Nhân hạch ma trận mảng A gồm m hàng và k cột, ma trận B gồm k hàng và n cột

$$C_{m \times n} = A_{m \times k} * B_{k \times n}$$

$$C = (c_{ij})_{m \times n}, c_{ij} = \sum_{t=0}^{k-1} a_{it} * b_{tj}$$
$$i = 0, 1, \dots, m - 1,$$
$$j = 0, 1, \dots, n - 1$$



# Ma trận tích C = AB



```
1 #include<stdio.h>
2 const MAX = 100;
3 int main(){
4     int A[MAX][MAX], B[MAX][MAX], C[MAX][MAX], m, k, n, i, j, t;
5     printf("Nhập m = "); scanf("%d", &m);
6     printf("Nhập k = "); scanf("%d", &k);
7     printf("Nhập n = "); scanf("%d", &n);
8     puts("Nhập ma trận A:");
9     for(i = 0; i < m; i++)
10    {
11        for(j = 0; j < k; j++)
12        {
13            printf("A[%d][%d] = ", i, j);
14            scanf("%d", &A[i][j]);
15        }
16    puts("Nhập ma trận B:");
17    for(i = 0; i < k; i++)
18    {
19        for(j = 0; j < n; j++)
20        {
21            printf("B[%d][%d] = ", i, j);
22            scanf("%d", &B[i][j]);
23        }
24    }
```

```
25    {
26        puts("Ma trận C = A*B:");
27        for(i=0; i<m; i++)
28        {
29            for(j=0; j<n; j++)
30            {
31                C[i][j] = 0;
32                for(t = 0; t < k; t++)
33                {
34                    C[i][j] += A[i][t]*B[t][j];
35                }
36                printf("%4d", C[i][j]);
37            }
38            printf("\n");
39        }
40    }
41    return 0;
42 }
```



# Bài tập về Ma trận



Nhập M, N ( $M, N < 30$ ) và một ma trận A kích thước  $M \times N$ .  
Đưa ma trận ra màn hình

- Tìm hàng/cột có tổng các phần tử lớn nhất
- Tìm số lớn nhất/nhỏ nhất và vị trí trong ma trận
- Đưa ra ma trận S cùng kích thước thỏa mãn

$$S_{ij} = \begin{cases} 1 & \text{nếu } A_{ij} > 0 \\ 0 & \text{nếu } A_{ij} = 0 \\ -1 & \text{nếu } A_{ij} < 0 \end{cases}$$



# Chương 4: Mảng, con trỏ và xâu ký tự



## 1. Mảng

- Khái niệm
- Khai báo và sử dụng
- Các thao tác thường gặp

## 2. Con trỏ

- Khái niệm và cách khai báo
- Toán tử địa chỉ (&), toán tử nội dung (\*)
- Phép toán trên con trỏ
- Con trỏ và mảng

## 3. Xâu ký tự

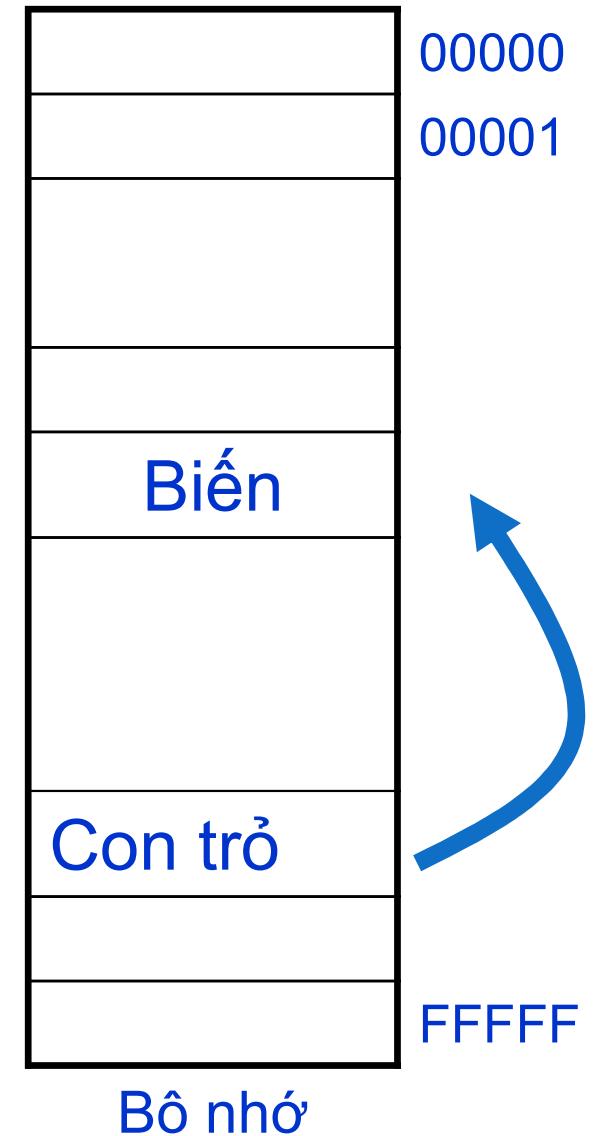
- Khái niệm, khai báo và sử dụng
- Các hàm xử lý ký tự và xâu ký tự
- Mảng xâu ký tự



# Giới thiệu



- Là một khái niệm “*mạnh*” trong C
  - Cho phép tính toán trên con trỏ
  - Sử dụng con trỏ hàm
- Cho phép truy nhập gián tiếp tới một đối tượng có địa chỉ (*biến, hàm*)
  - Truy nhập trực tiếp → thông qua tên





# Địa chỉ



- Bộ nhớ gồm dãy các ô nhớ
  - Mỗi ô nhớ là một byte
  - Mỗi ô nhớ có một địa chỉ riêng
- Các biến trong chương trình được lưu tại vùng nhớ nào đó trong bộ nhớ
- Khi khai báo biến, tùy thuộc vào kiểu, biến sẽ được cấp một số ô nhớ liên tục nhau
  - VD: Biến `int` được cấp 4 bytes, `float` được cấp 4 bytes,..
  - Địa chỉ của biến, là địa chỉ của byte đầu tiên trong số các byte được cấp
  - Khi gán giá trị cho biến, nội dung các byte cung cấp cho biến sẽ thay đổi



# Địa chỉ → Ví dụ



Địa chỉ của một biến là địa chỉ byte nhớ đầu tiên được cung cấp cho biến để lưu trữ dữ liệu

Ví dụ:

```
#include<stdio.h>
int main(){
    int n = 1000, m = 2000, p = 3000;
    printf("Dia chi cua n = %d: %x",n, &n);
    printf("\nDia chi cua m = %d: %x",m, &m);
    printf("\nDia chi cua p = %d: %x",p, &p);
    return 0;
}
```

```
Dia chi cua n = 1000: 61fe9c
Dia chi cua m = 2000: 61fe98
Dia chi cua p = 3000: 61fe94
-----
```



# Địa chỉ → Ví dụ



Địa chỉ của một biến là địa chỉ byte nhớ đầu tiên được cung cấp cho biến để lưu trữ dữ liệu

```
int n;  
float x;  
char a[4];
```

```
n = 1000; //03E8(16)  
x = 9.6875; //411B0000(16) theo IEEE752/85, 32 bit  
for(i=0; i<4; i++)  
    a[i] = 4*i+1;
```

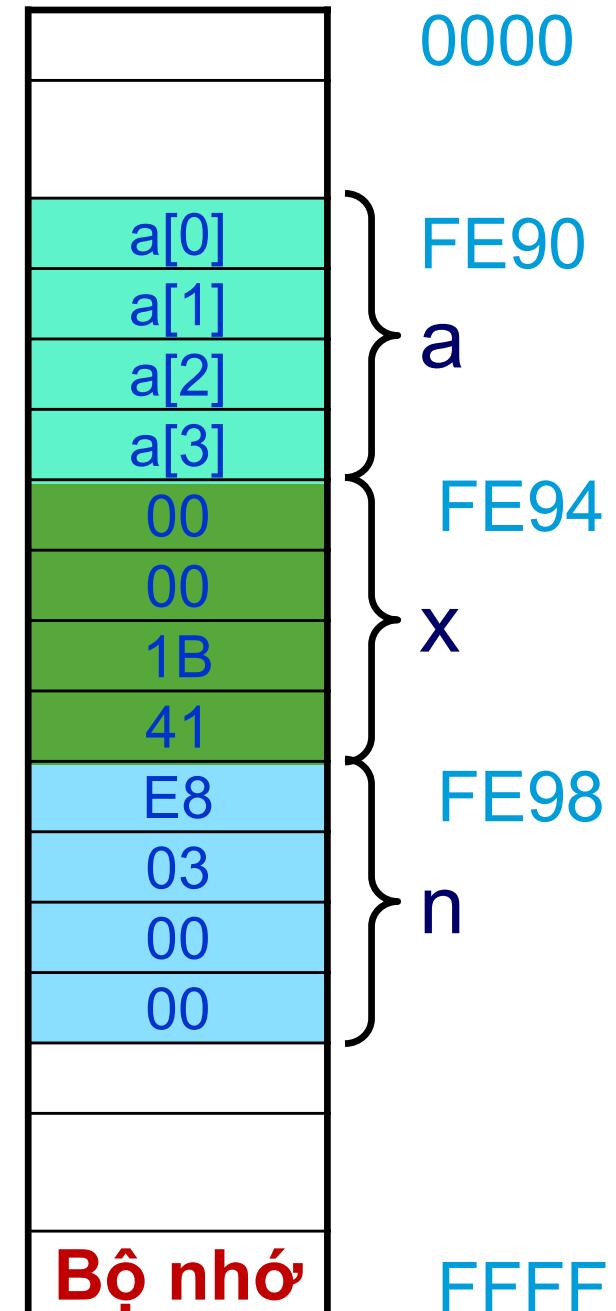


# Địa chỉ → Ví dụ



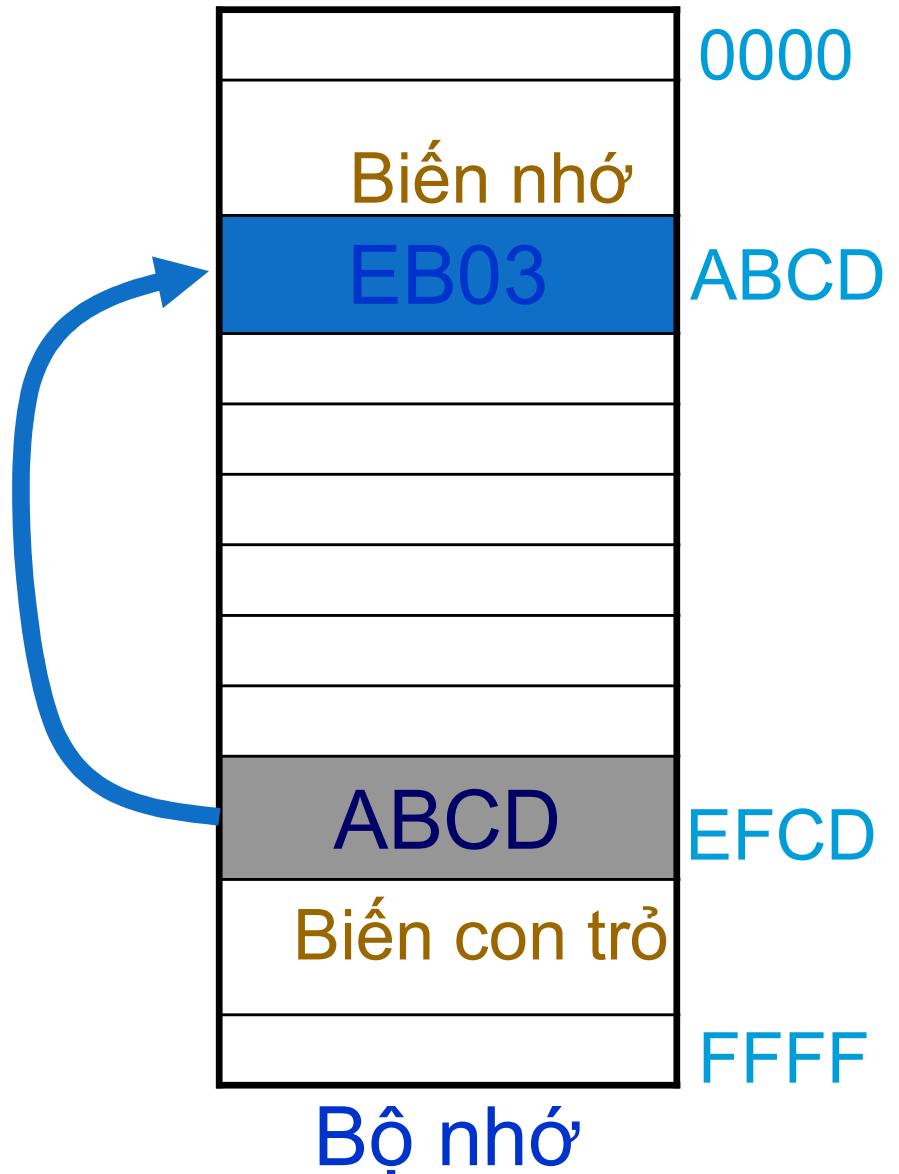
```
#include<stdio.h>
int main(){
    int n = 1000;
    float x = 9.6875;
    printf("Dia chi cua n = %d: %x",n, &n);
    printf("\nDia chi cua x = %.4f: %x",x, &x);
    char a[4];
    int i;
    for (i = 0;i<4;i++)
    {
        a[i] = 4*i+1;
        printf("\nDia chi cua a[%d] = %d: %x",i, a[i], &a[i]);
    }
    return 0;
}
```

```
Địa chỉ của n = 1000: 61fe98
Địa chỉ của x = 9.6875: 61fe94
Địa chỉ của a[0] = 1: 61fe90
Địa chỉ của a[1] = 5: 61fe91
Địa chỉ của a[2] = 9: 61fe92
Địa chỉ của a[3] = 13: 61fe93
-----
```



# Con trỏ

- Con trỏ là một biến mà giá trị của nó là địa chỉ của một vùng nhớ
  - Vùng nhớ này có thể dùng để chứa các biến có kiểu cơ bản (nguyên, thực, ký tự,...) hay có cấu trúc (mảng, bản ghi,...)
- Con trỏ dùng “trỏ tới” một biến nhớ
  - Có thể trỏ tới một hàm
  - Có thể trỏ tới con trỏ khác





# Con trả →Khai báo



Kiểu \*Tên;

- ❖ Tên: Tên của một biến con trả
- ❖ Kiểu: Kiểu của biến mà con trả “Tên” trả tới
  - Giá trị của con trả có thể thay đổi được
    - Trả tới các biến khác nhau, có cùng kiểu
  - Kiểu biến mà con trả trả tới không thay đổi được
    - Muốn thay đổi phải thực hiện “ép kiểu”
- ❖ Ví dụ:
  - int \*pi; //Con trả, trả tới một biến kiểu nguyên
  - char \*pc; //Con trả, trả tới một biến kiểu ký tự

- ❖ Ký hiệu: &
- ❖ Là toán tử một ngôi, trả về địa chỉ của biến

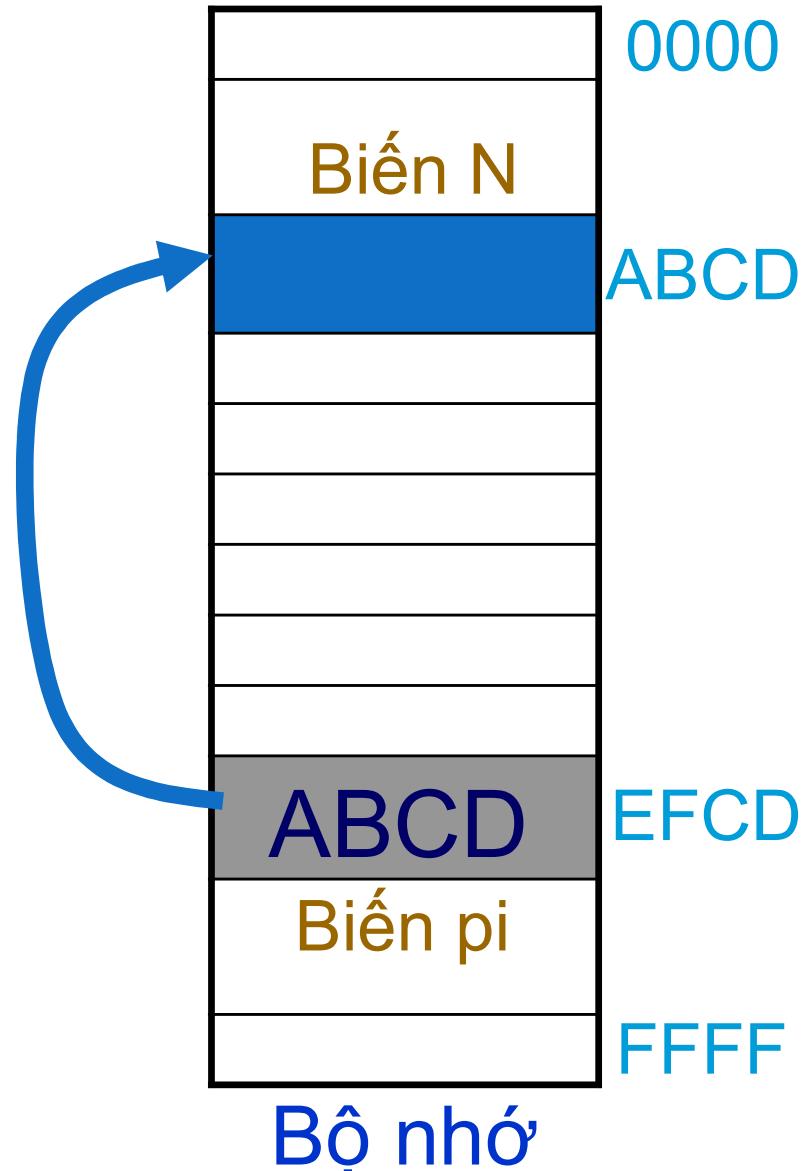
Địa chỉ biến có thể được gán cho một con trỏ, trỏ tới đối tượng cùng kiểu

- ❖ Ví dụ

```
short int N; // &N→ ABCD
```

```
short int *pi;
```

```
pi = &N; // pi←ABCD
```



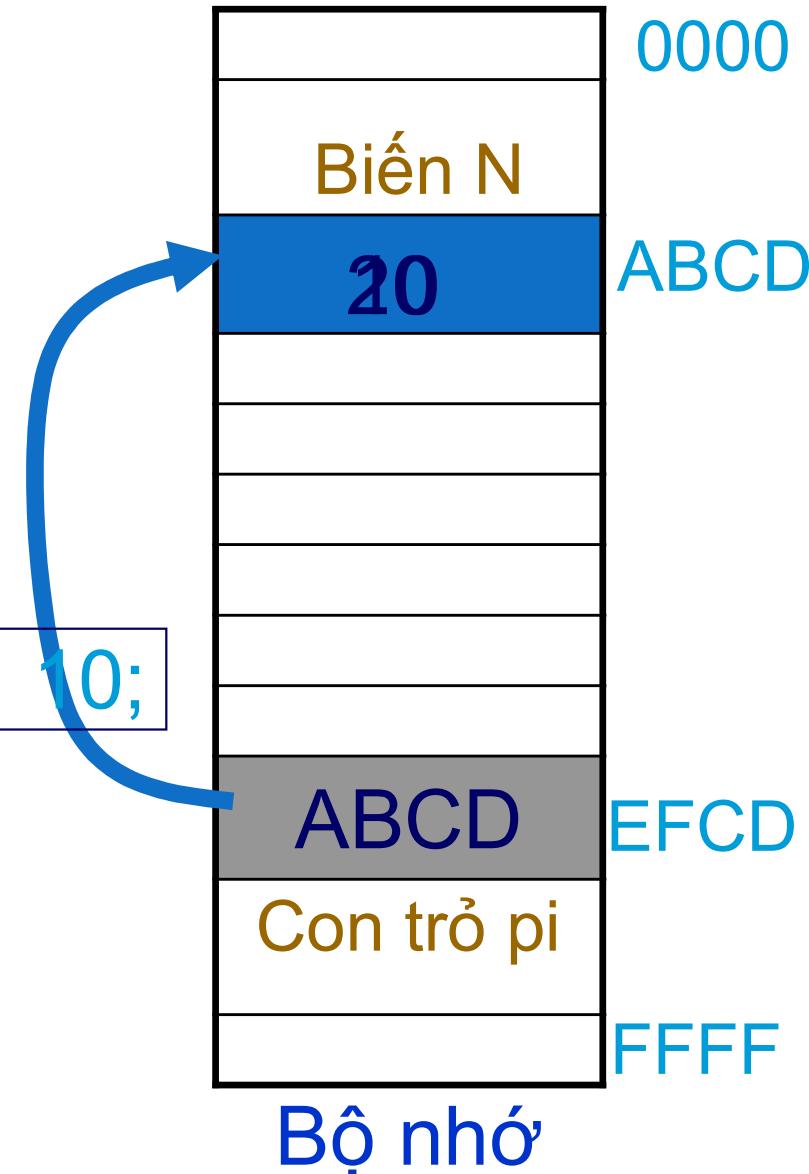


# Toán tử nội dung (\*)



- ❖ Ký hiệu: \*
  - ❖ Là toán tử một ngôi, trả về giá trị (nội dung) của vùng nhớ mà con trỏ đang trỏ tới

- ❖ **Ví dụ**
  - short int N;
  - short int \* pi;
  - pi = &N; }
  - N = 10; //Vùng nhớ mà pi trả tới mang giá trị 10; Vậy \*pi=10
  - \*pi = 20; // Vùng nhớ pi trả tới được gán giá trị 20; Vậy N= 20





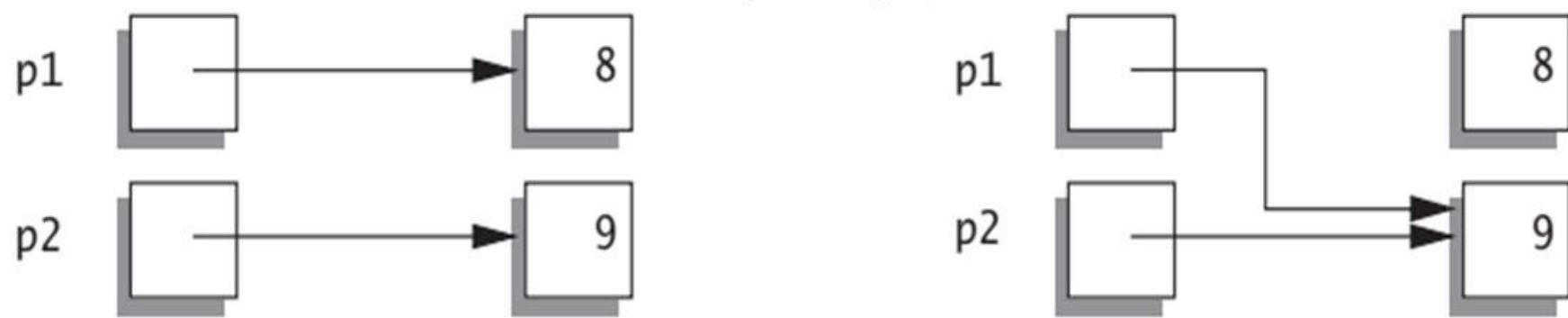
# Gán giá trị cho con trỏ



- ❖ Con trỏ được gán địa chỉ của một biến
    - Biển cùng kiểu với kiểu mà con trỏ trả tới  
Nếu không, cần phải ép kiểu
  - ❖ Con trỏ được gán giá trị của con trỏ khác
    - Hai con trỏ sẽ trả tới cùng một biến (do cùng địa chỉ)
    - Hai con trỏ nên cùng kiểu trả đến  
Nếu không, phải ép kiểu
  - ❖ Con trỏ được gán giá trị NULL
- Ví dụ:
- ```
int *p;  
p = 0;
```
- ❖ Gán nội dung vùng nhớ 2 con trỏ trả tới.
- Ví dụ:
- ```
int *p1, *p2;  
*p1 = *p2;
```



# Gán giá trị cho con trỏ



Trước \*p1 = \*p2; Sau





# Ví dụ



```
#include <stdio.h>
int main(){
    int N = 8, M = 9;
    int *p1 = &N;
    int *p2 = &M;
    *p1 = *p2;
    printf("%d %d", *p1, *p2);
}
```

9 9

```
#include <stdio.h>
int main(){
    int N = 8, M = 9;
    int *p1 = &N;
    int *p2 = &M;
    p1 = p2;
    printf("%d %d", *p1, *p2);
}
```

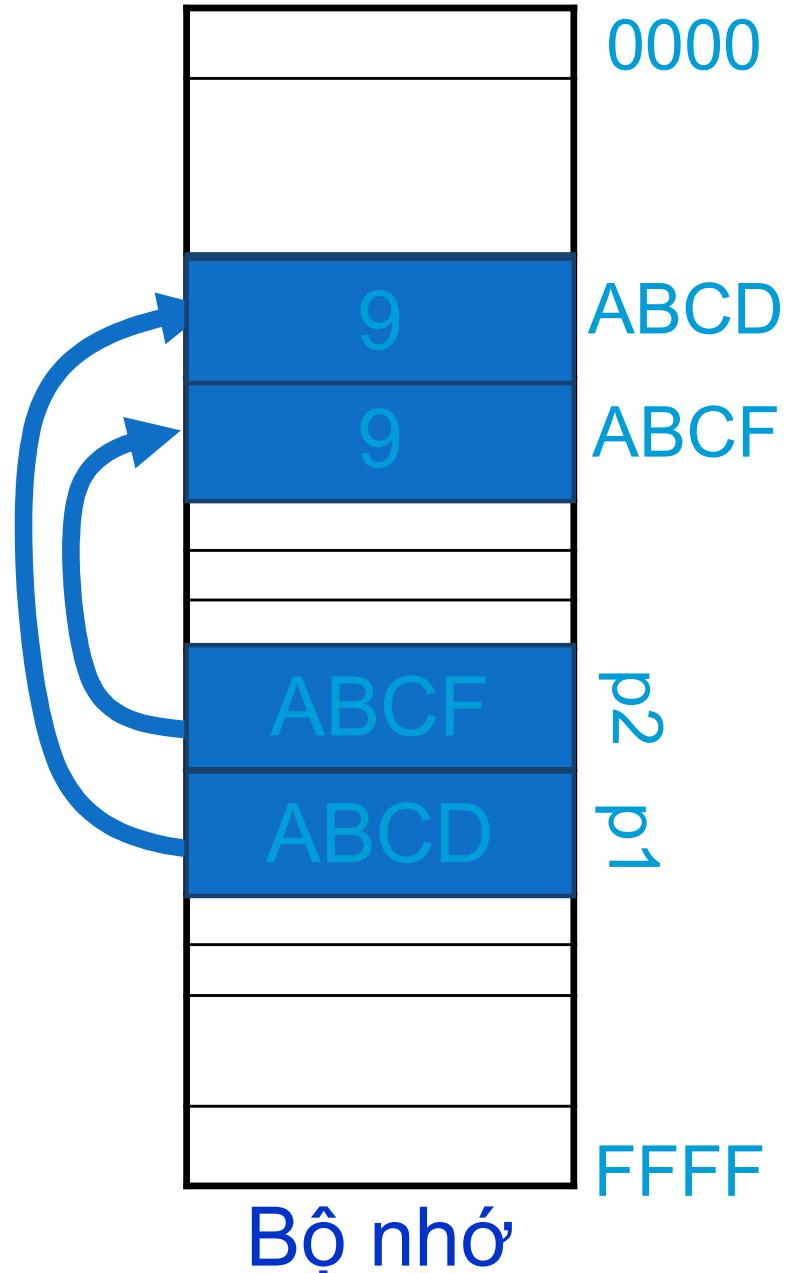
9 9



# Ví dụ → Trường hợp 1



```
#include <stdio.h>
int main(){
    int N = 8, M = 9;
    int *p1 = &N;
    int *p2 = &M;
    *p1 = *p2;
    printf("%d %d", *p1, *p2);
}
```



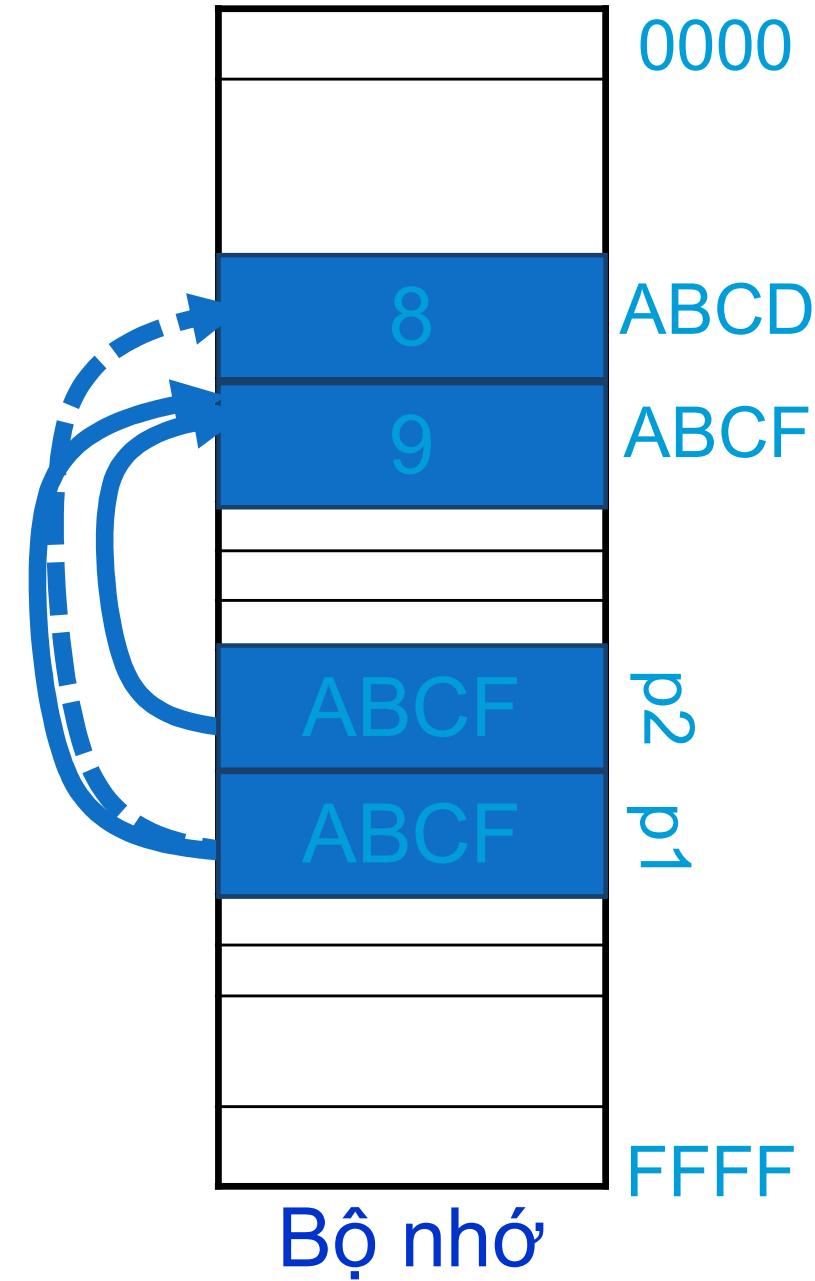


# Ví dụ → Trường hợp 2



```
#include <stdio.h>

int main(){
    int N = 8, M = 9;
    int *p1 = &N;
    int *p2 = &M;
    p1 = p2;
    printf("%d %d", *p1, *p2);
}
```





# Các phép toán trên con trỏ



- Cộng con trỏ với một số nguyên
  - Kết quả: Con trỏ cùng kiểu
- Trừ con trỏ với một số nguyên
  - Kết quả: Con trỏ cùng kiểu
- Trừ 2 con trỏ cùng kiểu cho nhau
  - Kết quả: Một số nguyên
  - Khoảng cách giữa 2 con trỏ được đo bằng số phần tử thuộc kiểu dữ liệu mà con trỏ trỏ tới



# Các phép toán trên con trỏ → Ví dụ



- int N=1000, M=2000, P=3000;
- int \* p1 = &P, \*p2 = &N;

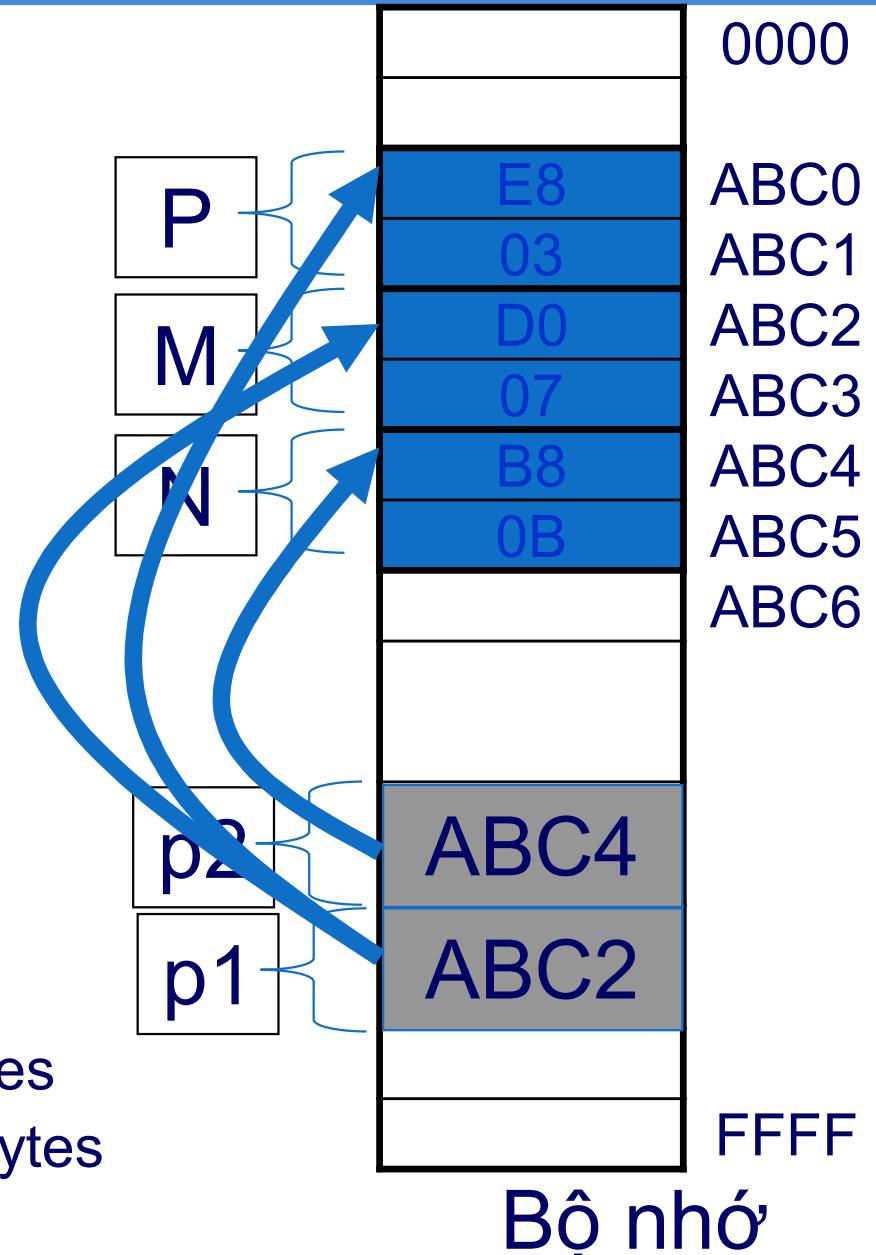
$p1 - p2 \rightarrow -2$

$* (p2-1) \rightarrow 2000$

$* ++ p1 \rightarrow 2000$

Ghi chú:

- Kiểu int, các phần tử cách nhau 4 bytes
- Kiểu float, các phần tử cách nhau 4 bytes





# Mỗi quan hệ giữa con trỏ và mảng một chiều



- ❖ Nếu T là tên một mảng  $\Rightarrow$  T là một con trỏ hằng chứa địa chỉ của phần tử đầu tiên của mảng T ( $\&T[0]$ )
  - Không tồn tại phép tính trên tên mảng, hoặc gán giá trị cho tên mảng (VD: T=...; T++)
- ❖ Có thể sử dụng một con trỏ để duyệt mảng nếu nó được gán giá trị bằng địa chỉ của mảng (địa chỉ của phần tử đầu tiên)



# Ví dụ



```
❖ int A[10];  
❖ int *p = A; // int *p = &A[0]
```

```
for(i = 0; i < 10; i ++)  
    printf("%d", *(p + i));
```

```
for(i = 0; i < 10; i ++)  
    printf("%d", p[i]);
```

```
for(i = 0; i < 10; i ++)  
    printf("%d", *(p++));
```



# Con trả void



void \* Tên\_con\_trả

- ❖ Là một con trả đặc biệt: con trả tới dữ liệu không định kiểu.
- ❖ Có thể nhận giá trị là địa chỉ của một biến có kiểu dữ liệu bất kỳ
- ❖ Ví dụ:

```
void * p, *q;  
int n; float x;  
p = &n; q= &x; //← Các câu lệnh hợp lệ
```



# Câu hỏi 1



```
#include<stdio.h>
int main(){
    int a = 3,*p;
    p = &a;
    printf("%d\n",a* *p *a + *p);
    return 0;
}
```

30



## Câu hỏi 2



```
#include<stdio.h>
int main(){
    int a[2][2][2] = {10, 2, 3, 4, 5, 6, 7, 8};
    int *p, *q;
    p = &a[1][1][1];
    q = (int *)a;
    printf("%d, %d\n", *p, *(q+4));
    return 0;
}
```

8, 5



# Chương 4: Mảng, con trỏ và xâu ký tự



## 1. Mảng

- Khái niệm
- Khai báo và sử dụng
- Các thao tác thường gặp

## 2. Con trỏ

- Khái niệm và cách khai báo
- Toán tử địa chỉ (&), toán tử nội dung (\*)
- Phép toán trên con trỏ
- Con trỏ và mảng

## 3. Xâu ký tự

- Khái niệm, khai báo và sử dụng
- Các hàm xử lý ký tự và xâu ký tự
- Mảng xâu ký tự



# Khái niệm xâu ký tự



'T'	'i'	' n'	' '	'h'	'o'	'c'	'\0'
-----	-----	------	-----	-----	-----	-----	------

- Xâu ký tự (string) là một dãy các ký tự viết liên tiếp nhau
  - Độ dài xâu là số ký tự có trong xâu
  - Xâu không có ký tự nào: Xâu rỗng
- Ví dụ: "Tin hoc", "String"
- Lưu trũ: kết thúc xâu bằng ký tự '\0' hay NULL (mã ASCII là 0)



# Khái niệm xâu ký tự → Lưu ý



- Xâu kí tự >< mảng kí tự
  - Tập hợp các kí tự viết liên tiếp nhau
    - Truy nhập một phần tử của xâu ký tự (*là một ký tự*) giống như truy nhập vào một phần tử của mảng:  
Tên[Chỉ số]
    - Xâu kí tự có kí tự kết thúc xâu, mảng kí tự không có kí tự kết thúc xâu
  - Xâu kí tự độ dài 1 >< kí tự ("A" =='A' ?)
    - 'A' là 1 kí tự, được lưu trữ trong 1 byte
    - "A" là 1 xâu kí tự, ngoài kí tự 'A' còn có kí tự '\0' => được lưu trữ trong 2 byte



# Khai báo



**char tên\_xâu [số\_kí\_tự\_tối\_đa];**

- Để lưu trữ một xâu có n kí tự chúng ta cần một mảng có kích thước n+1
  - Phần tử cuối cùng chứa ký tự NULL

## Ví dụ

- Để lưu trữ xâu “Tin hoc” chúng ta phải khai báo xâu có số phần tử tối đa ít nhất là 8

char str[8] = "Tin hoc";



# Truy nhập phần tử của xâu



Giống như truy nhập tới một phần tử của mảng ký tự:

tên\_xâu [chỉ\_số\_của\_kí\_tự]

Ví dụ: char Str[10] = "Tin hoc";

T	i	n		h	o	c	\0	?	?
---	---	---	--	---	---	---	----	---	---

Str[0] → 'T'

Str[8] → ?

Str[3] → ' '

Str[9] → ?

Str[7] → '\0'



# Truy nhập phần tử của xâu



Ví dụ: char Str[10] = "Tin hoc";

T	i	n		h	o	c	\0		
---	---	---	--	---	---	---	----	--	--

Str[3] = '-';

Str[8] = '1' ;

Str[7] = ' ';

Str[9] = '\0';

T	i	n	-	h	o	c		1	\0
---	---	---	---	---	---	---	--	---	----

Str: Tin-hoc 1



# Ví dụ: Nhập xâu và đếm số ký tự '\*'



```
#include <stdio.h>
int main(){
    char str[100];
    int d = 0, i = 0;
    printf("Nhập xâu ký tự: "); gets(str);
    while(str[i]!='\0'){
        if(str[i]=='*')
            d++;
        i++;
    }
    printf("Kết quả: %d",d);
    return 0;
}
```

Tính chiều dài của xâu  
d = 0;  
while(str[d] != '\0')  
 d++;



# Ví dụ: Nhập xâu và đếm số ký tự 'a'



```
#include <stdio.h>
int main(){
    char str[100];
    int d = 0, i = 0;
    printf("Nhập xâu ký tự: "); gets(str);
    while(str[i]!='\0'){
        if(str[i]=='a')
            d++;
        i++;
    }
    printf("Kết quả: %d", d);
    return 0;
}
```



# Ví dụ: Nhập câu và đưa ra dưới dạng cột



```
#include <stdio.h>
int main(){
    char S[100];
    int i = 0;
    printf("Nhập xau: "); gets(S);
    while(S[i]!='\0')
    {
        //32 là mã ASCII của phím Space
        if(S[i] != 32 && S[i+1]==32)
            printf("%c\n", S[i]);
        else
            if(S[i]!= 32) printf("%c",S[i]);
        i++;
    }
    return 0;
}
```

Đếm số từ, nếu các từ được cách nhau bởi dấu phân cách



# Các hàm xử lý ký tự



Tệp tiêu đề : ctype.h

#include <ctype.h>



# Các hàm xử lý ký tự → Chuyển đổi chữ hoa/thường



- int toupper(char ch):
  - Chuyển kí tự thường thành kí tự hoa  
toupper('a') => 'A'
- int tolower(char ch)
  - Chuyển kí tự hoa thành kí tự thường  
tolower('B') => 'b'

## Ví dụ

do{

.....

```
printf("Tiep tuc <C/K>?");  
fflush(stdin);  
}while(toupper(getchar()) !='K');
```



# Các hàm xử lý ký tự → Kiểm tra chữ hoa/thường



- int **islower(char ch)**
  - Kiểm tra chữ thường:
    - Hàm trả về giá trị khác 0 nếu ch là chữ thường, ngược lại trả về 0
    - Ví dụ: printf("%d ", **islower('A')**);  $\Rightarrow 0$
  - int **isupper(char ch)**:
    - Kiểm tra chữ hoa:
      - Hàm trả về giá trị khác 0 nếu ch là chữ hoa, ngược lại trả về 0
      - Ví dụ: printf("%d ", **isupper('A')**);  $\Rightarrow \neq 0$  (1 !?)



# Các hàm xử lý ký tự → Kiểm tra chữ cái/chữ số



- **int isalpha(char ch):**

- Kiểm tra kí tự trong tham số có phải chữ cái không ('a'...'z','A',..'Z'). Hàm trả về khác 0 nếu đúng, ngược lại trả về giá trị bằng 0
  - Ví dụ: `printf("%d ",isalpha('A'));` ⇒ ≠ 0 (1 !?)

- **int isdigit(char ch):**

- Kiểm tra kí tự trong tham số có phải chữ số ('0','1',..'9') không. Hàm trả về khác 0 nếu đúng, ngược lại trả về giá trị bằng 0
  - Ví dụ: `printf("%d ",isdigit('A'));` ⇒ 0



# Khái niệm xâu ký tự → Kiểm tra ký tự đặc biệt



- **int iscntrl(char ch)**
  - Kiểm tra ký tự điều khiển (0-31).
  - Hàm trả về khác 0 nếu đúng, ngược lại trả về giá trị bằng 0
- **int isspace(char ch)**
  - Kiểm tra ký tự dấu cách (mã 32), xuống dòng ('\n' 10), đầu dòng ('\r' 13), tab ngang ('\t' 9), tab dọc ('\v' 11).
  - Hàm trả về khác 0 nếu đúng, ngược lại trả về giá trị bằng 0



# Ví dụ: Nhập xâu, chuyển thành xâu chữ hoa



```
#include <stdio.h>
#include <ctype.h>
int main(){
    char S[50];
    int i = 0;
    printf("Nhập một xâu: "); gets(S);
    printf("\nXâu ban đầu: %s.",S);
    while(S[i]!='\0'){
        S[i] = toupper(S[i]);
        i = i + 1;
    }
    printf("\nXâu kết quả: %s",S);
    return 0;
}
```

```
Nhap mot xau: hoc nua, hoc nua, hoc mai!
Xau ban dau: hoc nua, hoc nua, hoc mai!.
Xau ket qua: HOC NUA, HOC NUA, HOC MAI!
-----
Process exited after 23.1 seconds with return value 0
Press any key to continue . . .
```



Ví dụ: Nhập xâu và đếm từ, các từ được phân cách bởi dấu trắng



# Ví dụ: Nhập xâu và đếm từ, các từ được phân cách bởi dấu trắng



```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
int main(){
    char s[100];
    int d = 0, i = 0;
    printf("Nhập xâu ký tự: "); gets(s);
    if(s[0] == '\0')
        printf("Xau rong!");
    else{
        if(!isspace(s[0]))
            d = 1;
        i = 1;
        while(s[i] != '\0'){
            if(isspace(s[i-1]) && (!isspace(s[i])))
                d++;
            i++;
        }
        printf("Kết quả: %d", d);
    }
    return 0;
}
```



# Ví dụ: Nhập xâu và đếm từ, các từ được phân cách bởi dấu trắng.



```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
int main(){
    char s[100];
    int d = 0, i = 0;
    printf("Nhập xau ky tu: "); gets(s);
    for (i = 0;s[i]!='\0';i++)
        if(s[i]!=' '&&s[i+1] == ' ' || s[i] != ' '&&s[i+1] == '\0')
            d++;
    if (d==0)
        printf("Xau rong!");
    else
        printf("Ket qua: %d",d);
    return 0;
}
```



# Các hàm xử lý xâu ký tự



## Vào/ra xâu ký tự

- Tệp tiêu đề: stdio.h
- Nhập xâu kí tự
  - gets(tên\_xâu);
  - scanf("%s",&tên\_xâu);
- Hiển thị xâu kí tự
  - puts(tên\_xâu);
  - printf("%s",tên\_xâu);

Sự khác nhau giữa gets và scanf?



# Lưu ý:



- ❖ Hàm gets không kiểm tra kích thước xâu được nhập

```
#include <stdio.h>
#include <ctype.h>
int main(){
    char s[10];
    int i = 0;
    printf("Nhập xâu ký tự: "); gets(s);
    printf("Xau da nhap: %s",s);
    return 0;
}
```

Khai báo xâu ký tự có tối đa 10 ký tự, nhưng có thể nhập nhiều hơn 10 ký tự với gets()

- ❖ Giải pháp: dùng fgets(str, so\_ky\_tu\_toida, stdin);



# Lưu ý:



❖ Giải pháp: dùng fgets(str, so\_ky\_tu\_toida, stdin);

❖ Ví dụ:

```
#include <stdio.h>
int main(){
    char s[10], w[10];
    printf("Nhập xau ky tu s: "); gets(s);
    printf("Nhập xau ky tu w: "); fgets(w,10,stdin);
    printf("\nXau da nhap s: %s", s);
    printf("\nXau da nhap w: %s", w);
    return 0;
}
```



# Các hàm xử lý xâu ký tự



Tệp tiêu đề: string.h

#include <string.h>

## Chú ý:

```
char str[100] = "Hello world";
```

```
char * p = str;
```

- p là con trỏ, trỏ tới mảng các ký tự/xâu ký tự
- p+6: (Phép tính toán trên con trỏ), cũng là xâu ký tự. p+6 trỏ tới xâu "world".
- Xâu ký tự, có thể được khai báo **char \***



# Các hàm xử lý xâu ký tự



size\_t **strlen**(const char \* xâu)

- Trả về độ dài xâu

```
printf("%d ",strlen("Hello world")); //11
```

char \* **strcpy**(char \* đích, const char \* nguồn)

- Sao chép nội dung xâu *nguồn* vào xâu *đích*, trả về giá trị xâu nguồn

```
char Str[20];
```

```
printf("%s",strcpy(Str,"Hello")); //Hello
```

```
printf("%s", Str); //Hello
```

Chú ý: Phép gán Str = "Hello" là không hợp lệ



# Các hàm xử lý xâu ký tự



int **strcmp**(const char \* xâu\_1, const char \* xâu\_2)

- So sánh hai xâu.
- Trả về giá trị 0 nếu hai xâu giống nhau;
- Giá trị < 0: xâu\_1 < xâu\_2
- Giá trị >0: xâu\_1 > xâu\_2

## Ví dụ

```
char Str[20];
```

```
strcpy(Str, "hello");
```

```
printf("%d", strcmp(Str, "hello")); → 0
```

```
printf("%d", strcmp(Str, "hello! ")); → -1 (!?)
```

```
printf("%d", strcmp(Str, "Hello")); → 1 (!?)
```



# Các hàm xử lý xâu ký tự



char \***strcat**(char \*đích, const char \*nguồn)

- Ghép nối xâu nguồn vào ngay sau xâu đích, trả lại xâu kết quả

## Ví dụ

```
char Str[20];
```

```
strcpy(Str,"Hello ");
```

```
printf("%s ",strcat(Str,"world")); ⇒ Hello world
```

```
printf("\n%s",Str); ⇒ Hello world
```



# Các hàm xử lý xâu ký tự



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

- Trả về con trỏ trỏ tới vị trí xuất hiện đầu tiên của ký tự c trong s. Nếu không có trả về con trỏ null

`strcpy(Str,"Hello world");`

`printf("%s ",strchr(Str,'o'));` ⇒ o world

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

- Trả về con trỏ trỏ tới vị trí xuất hiện đầu tiên của chuỗi s2 trong s1. Nếu không tồn tại, trả về con trỏ null

`printf("%s ",strstr(Str, "llo"));` ⇒ llo world



## Lưu ý:



- ❖ Tương tự như hàm `gets`, một số hàm xử lý xâu: `strcpy`, `strcat` cũng đã không dùng được đối với các trình biên dịch mới.
- ❖ Tuy nhiên trong phạm vi môn Tin cơ sở, chúng ta vẫn chấp nhận dùng các hàm trên với trình biên dịch phù hợp.



# Các hàm xử lý xâu ký tự (tiếp)



## Tệp tiêu đề: stdlib.h

- int **atoi**(const char \* str):
  - Chuyển một xâu kí tự thành một số nguyên tương ứng
  - Ví dụ: atoi("1234") → 1234
- int **atol**(const char \* str):
  - Chuyển xâu kí tự thành số long int
- float **atof**(const char \* str):
  - Chuyển xâu kí tự thành số thực
  - Ví dụ: atof("123.456E-2") → 1.23456
- Thất bại cả 3 hàm: trả về 0



# Các hàm xử lý xâu ký tự (tiếp)



## Tập tiêu đề <stdio.h>

- ❖ int sscanf (const char \* s, const char \* format, ...);
  - Hoạt động: giống với hàm scanf, chỉ khác là **thay vì đọc từ bàn phím** thì sẽ **đọc từ xâu ký tự**

```
#include<stdio.h>

int main(){
    char str[10] = "10 20 30";
    int a, b, c;

    sscanf(str, "%d %d %d", &a, &b, &c);
    printf("a = %d, b = %d, c = %d", a, b, c);
}
```



# Các hàm xử lý xâu ký tự (tiếp)



## Tập tiêu đề <stdio.h>

❖ int sprintf (char \* s, const char \* format, ...);

- Hoạt động: giống với hàm printf, chỉ khác là thay vì in ra màn hình thì sẽ “in” vào xâu ký tự.

```
#include<stdio.h>

int main(){
    char str[10];
    int a = 10;
    float b = 20;
    char str2[10] = "abc";

    sprintf(str, "%d %f %s", a, b, str2);
    printf("str = %s", str);
}
```



# Ví dụ 1: Nhập 2 xâu, cho biết số lần xuất hiện xâu 2 trong xâu 1.



Nhap xau thu nhat: abcabcabcdef gh

Nhap xau thu hai: abc

Xau "abc" hien trong xau "abcabcabcdef gh" 3 lan

-----

Nhap xau thu nhat: hoc nua hoc mai hoc suot doi

Nhap xau thu hai: hoc

Xau "hoc" hien trong xau "hoc nua hoc mai hoc suot doi" 3 lan

-----



# Ví dụ 1: Nhập 2 xâu cho biết số lần xuất hiện xâu 2 trong xâu 1



```
#include <stdio.h>
#include <string.h>
int main(){
    int d = 0;
    char S1[100], S2[50], *p;
    printf("Nhập xâu thu nhất: "); gets(S1);
    printf("Nhập xâu thu hai: "); gets(S2);
    p = strstr(S1,S2);
    while(p != NULL){
        d = d + 1;
        p = strstr(p+1,S2); //vi tri tim kiem ke tiep
    }
    printf("Xâu \"%s\" hien trong xau \"%s\" %d lan",S2,S1,d);
    return 0;
}
```

```
Nhap xau thu nhat: hoc nua hoc mai hoc suot doi
Nhap xau thu hai: hoc
Xau "hoc" hien trong xau "hoc nua hoc mai hoc suot doi" 3 lan
```



# Ví dụ 2: Kiểm tra xâu đối xứng



```
Nhap vao xau ki tu: abccba  
Xau doi xung!  
-----
```

```
Nhap vao xau ki tu: abcdeghf  
Xau khong doi xung!  
-----
```



# Ví dụ 2: Kiểm tra xâu đối xứng



```
#include<stdio.h>
#include<string.h>
int main(){
    char s[50];
    printf("Nhập vào xâu ki tự: "); gets(s);
    int i, n = strlen(s);
    for(i = 0; i<n/2; i++)
        if(s[i] != s[n-1-i])
            break;
    if(i==n/2)
        printf("Xâu đối xứng!");
    else
        printf("Xâu không đối xứng!");
    return 0;
}
```

Nhập vào xâu ki tự: abccba  
Xâu đối xứng!  
-----

Nhập vào xâu ki tự: abcdeghf  
Xâu không đối xứng!  
-----



# Ví dụ 3: Đảo ngược xâu ký tự



```
#include<stdio.h>
#include<string.h>
int main(){
    char s[100], c;
    printf("Nhập xau: "); gets(s);
    int i, n = strlen(s);
    for(i = 0; i<n/2; i++){
        c = s[i];
        s[i] = s[n-i-1];
        s[n-i-1] = c;
    }
    printf("Xau đảo ngược: %s",s);
    return 0;
}
```



# Ví dụ 4



Đếm số lần xuất hiện chữ cái trong xâu (không phân biệt hoa thường)

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
int main(){
    char s[20];
    int dem[26] = {};
    int i, n;
    puts("Nhập vào xâu ki tu:"); gets(s);
    n = strlen(s);
    for(i=0;i<n;i++)
        if(isalpha(s[i]))
            dem[tolower(s[i])-'a']++;
    for(i=0;i<26;i++)
        if(dem[i]!=0)
            printf("\nKi tu %c xuất hiện %d lần.", 'a'+i,dem[i]);
    return 0;
}
```



# Mảng xâu ký tự



- Xâu ký tự có thể là kiểu phần tử của mảng
- Khai báo  
char DS[100][30];  
Mảng có tối đa 100 phần tử, các phần tử là xâu có độ dài tối đa 30
- Sử dụng
  - Như một mảng bình thường
  - Mỗi phần tử mảng được sử dụng như một xâu ký tự



## Ví dụ



Nhập vào DSSV cho tới khi gấp tên rõng, in Danh sách vừa nhập.

```
Nhap DSSV (<100), go Enter de thoat..  
Ten sinh vien [1]: Hoang Mai  
Ten sinh vien [2]: Tran Anh  
Ten sinh vien [3]: Nguyen Hoang  
Ten sinh vien [4]: Mai Hong  
Ten sinh vien [5]:  
  
DS sinh vien vua nhap:  
Hoang Mai  
Tran Anh  
Nguyen Hoang  
Mai Hong  
-----
```



# Ví dụ: Nhập vào DSSV cho tới khi gặp tên rỗng, in DS



```
#include <stdio.h>
#include <string.h>
int main(){
    int i, n;
    char DS[100][30];
    printf("Nhập DSSV (<100), go Enter để thoát..\n");
    n = 0;
    do{
        printf("Tên sinh viên [%d]: ", n+1); gets(DS[n]);
        if(DS[n][0] != '\0') n++;
        //if(strcmp(DS[n], "") == 0) n++;
        //if(strlen(DS[n]) > 0) n++;
        else break;
        if(n==100) break;
    }while(1);
    printf("\nDS sinh viên vừa nhập:");
    for(i=0;i<n;i++)
        printf("\n%s", DS[i]);
    return 0;
}
```



# Ví dụ: Nhập vào DSSV cho tới khi gặp tên rỗng, in DS



```
#include <stdio.h>
#include <string.h>
int main(){
    int i, n;
    char DS[100][30];
    printf("Nhập DSSV (<100), go Enter để thoát..\n");
    for(n = 0; n < 100; n++){
        printf("Tên sinh viên [%d]: ", n+1); gets(DS[n]);
        if(DS[n][0] == '\0') break;
    }
    printf("\nDS sinh viên vừa nhập:");
    for(i=0;i<n;i++)
        printf("\n%s", DS[i]);
    return 0;
}
```



## Ví dụ



Nhập dãy (<100) xâu cho tới khi gấp xâu "\*\*\*". Đưa ra màn hình xâu có độ dài lớn nhất

```
Nhap xau thu [1]: Ha Noi
Nhap xau thu [2]: Thai Nguyen
Nhap xau thu [3]: Thai Binh
Nhap xau thu [4]: Ha Giang
Nhap xau thu [5]: Hoa Binh
Nhap xau thu [6]: ***
```

```
Xau dai nhat la: Thai Nguyen, co do dai: 11.
```

```
-----
```



# Ví dụ



Nhập dãy (<100) xâu cho tới khi gặp xâu "\*\*\*". Đưa ra màn hình xâu có độ dài lớn nhất (không tính xâu "\*\*\*")

```
#include <stdio.h>
#include <string.h>
int main(){
    int i, n = 0, d = 0;
    char DS[100][30], s[30] = "";
    do{
        printf("Nhập xau thu [%d]: ",n+1); gets(DS[n]);
        if(strcmp(DS[n],"***"))
            n = n + 1; //Khong tinh xau ***
        else
            break;
    }while(1);
    for(i = 0; i < n; i++)
        if(strlen(DS[i]) > d){
            d = strlen(DS[i]);
            strcpy(s,DS[i]);
        }
    printf("\nXau dai nhat la: %s, co do dai: %d.",s,d);
    return 0;
}
```



# Ví dụ



Nhập vào DS tên sinh viên, in ra DS đã sắp xếp.

```
So sinh vien: 3
```

```
Ten sinh vien [1]: Tran Mai
```

```
Ten sinh vien [2]: Hoang Hoa
```

```
Ten sinh vien [3]: Nguyen Ngan
```

```
DS sinh vien vua nhap:
```

```
Hoang Hoa
```

```
Nguyen Ngan
```

```
Tran Mai
```

```
-----
```



# Ví dụ: Nhập vào DS tên sinh viên, in ra DS đã sắp xếp.



```
#include <stdio.h>
#include <string.h>
int main(){
    int i, j, N;
    char DS[100][30], s[30];
    //Nhập DS sinh viên
    printf("So sinh vien: ");
    scanf("%d",&N);
    fflush(stdin);
    for(i = 0; i<N; i++){
        printf("Ten sinh vien [%d]: ",i+1) ;
        gets(DS[i]);
    }
    //So sánh theo họ tên
    for(i = 0;i<N-1; i++)
        for(j = i+1;j<N;j++)
            if(strcmp(DS[i],DS[j]) > 0){
                strcpy(s,DS[i]);
                strcpy(DS[i],DS[j]);
                strcpy(DS[j],s);
            }
}
```

```
//In danh sách đã sắp xếp
printf("\nDS sinh viên vừa nhập:");
for(i = 0;i<N;i++)
    printf("\n%s",DS[i]);
return 0;
}
```

```
So sinh vien: 3
Ten sinh vien [1]: Tran Mai
Ten sinh vien [2]: Hoang Hoa
Ten sinh vien [3]: Nguyen Ngan
```

```
DS sinh vien vua nhap:
Hoang Hoa
Nguyen Ngan
Tran Mai
-----
```



# Ví dụ



Nhập vào DS họ tên sinh viên, in ra DS đã sắp xếp theo tên.

```
So sinh vien: 3
Ho ten SV [1]: Bui Mai Ha
Ho ten SV [2]: Nguyen Anh
Ho ten SV [3]: Tran Thu Ha
```

```
DSSV sap xep theo ten:
Nguyen Anh
Bui Mai Ha
Tran Thu Ha
-----
```



# Ví dụ: Nhập vào DS họ tên sinh viên, in ra DS đã sắp xếp theo tên.



```
#include <stdio.h>
#include <string.h>
int main(){
    int i, j, N;
    char DS[100][30], str[30];
    //Nhập DS sinh viên
    printf("So sinh vien: ");
    scanf("%d",&N);
    fflush(stdin);
    for(i = 0; i<N; i++){
        printf("Ho ten SV [%d]: ",i+1);
        gets(DS[i]);
    }
    //Sắp xếp theo tên
    char ten_i[30],ten_j[30];
    for(i = 0; i<N-1; i++)
        for(j = i+1;j<N; j++){
            //trích ra từ cuối
            strcpy(ten_i,strrchr(DS[i],32));
            strcpy(ten_j,strrchr(DS[j],32));
            if(strcmp(ten_i,ten_j) > 0){
                strcpy(str,DS[i]);
                strcpy(DS[i],DS[j]);
                strcpy(DS[j],str);
            }
        }
    //In danh sách đã sắp xếp
    printf("\nDSSV sap xep theo ten:");
    for(i = 0;i<N;i++)
        printf("\n%s",DS[i]);
    return 0;
}
```



# Bài tập



1. Nhập vào 2 xâu S1, S2 và một số nguyên k. Hãy chèn xâu S1 vào S2 và đưa ra màn hình (*giả thiết xâu S2 được khai báo đủ lớn*)
2. Một văn bản gồm không quá 60 dòng, mỗi dòng không quá 80 ký tự. Hãy viết chương trình thực hiện nhập vào một văn bản, sau đó
  - a. Nhập vào xâu s và chỉ ra vị trí xuất hiện của xâu S trong văn bản nếu có.
  - b. Thay tất cả các chuỗi "hanoi" (nếu có) bằng chuỗi "HANOI"
  - c. Đếm xem trong văn bản có bao nhiêu từ (*các từ phân cách bởi dấu cách*)
  - d. Tính tần suất xuất hiện của các từ trong văn bản