



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

TIN HỌC ĐẠI CƯƠNG

Phần II: LẬP TRÌNH C

Nội dung chính

- Chương 1: Tổng quan về ngôn ngữ C
- Chương 2: Kiểu dữ liệu và biểu thức trong C
- Chương 3: Vào ra dữ liệu
- Chương 4: Cấu trúc điều khiển
- **Chương 5: Mảng, con trỏ và chuỗi ký tự**
- Chương 6: Cấu trúc
- Chương 7: Hàm
- Chương 8: Tập dữ liệu

Chương 5: Mảng, con trỏ và chuỗi ký tự

5.1. Mảng

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

5.2. 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

5.3. Chuỗi ký tự

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

Chương 5: Mảng, con trỏ và chuỗi ký tự

5.1. Mảng

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

5.2. 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

5.3. Chuỗi ký tự

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

Giới thiệu

Bài toán:

- Nhập điểm thi (số nguyên) môn Tin đại cương cho lớp gồm 50 sinh viên rồi đưa ra 10 sinh viên có điểm cao nhất.

Phương pháp: Điểm của mỗi sinh viên là 1 biến

- Tên biến là tên sinh viên
Ví dụ: `int An, Anh, Binh1, Binh2, Cuong,..... Van, Viet;`
- Tên biến dạng “*dx*” với x là chỉ số thứ tự của SV trong lớp
Ví dụ: `int d1, d2, d3,.....,d50;`

Nhận xét 1: Không hợp lý

- Có quá nhiều biến (*Điểm thi cho toàn trường.. !?*)
- Khó khăn cho các thao tác duyệt toàn bộ danh sách

Nhận xét 2: Các biến có chung ý nghĩa, tính chất

Giới thiệu

- Trong thực tế, thường gặp các đối tượng có tính chất chung:
 - Tháng trong năm
 - Điểm trung bình của sinh viên trong lớp
- Các đối tượng được nhóm lại dưới một tên
- Đối tượng được đặc trưng bởi **tên nhóm** và **thứ tự** trong nhóm
 - Tháng thứ 3 trong năm: Tháng 3
 - Sinh viên thứ 17 trong lớp:...
- Số thứ tự của đối tượng trong nhóm là **chỉ số phần tử**.

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

Ví dụ:

<d7>: Điểm thi THPTC của sinh viên thứ tự 7 trong lớp

Khai báo mảng

```
Kiểu_dữ_liệu Tên_Mảng[Kích_thước_mảng];
```

- Kiểu_dữ_liệu: kiểu của các phần tử trong mảng (*số nguyên, số thực, kí tự, chuỗi, mảng,...*)
- Tên_mảng: tên của mảng
- Kích_thước_mảng: số phần tử tối đa trong mảng

Ví dụ

```
int DiemTin[50]; // Khai báo mảng 50 phần tử có  
kiểu dữ liệu int
```

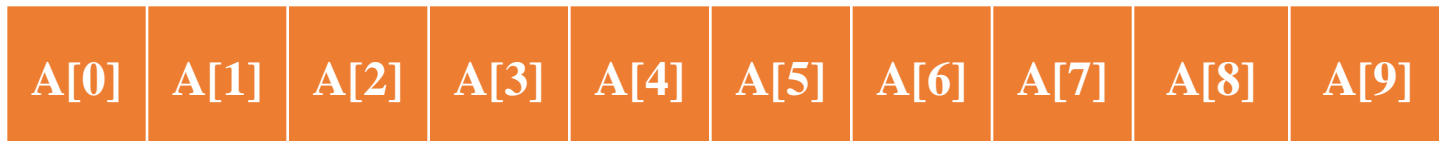
```
float A[10]; // Mảng 10 phần tử kiểu số thực
```


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 được cấp phát.

Ví dụ:

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



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

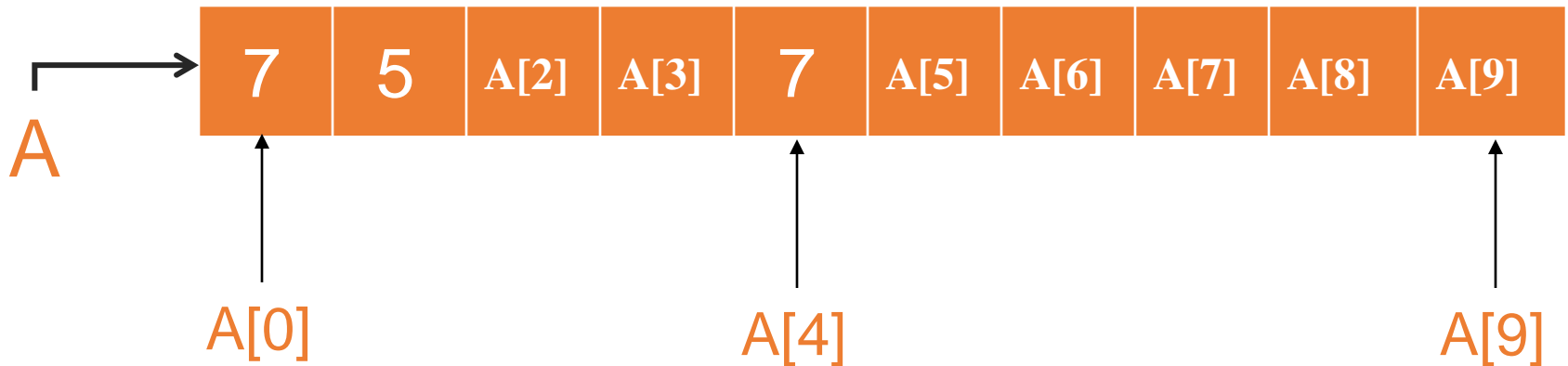
Truy nhập phần tử của mảng

- 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
 - Chỉ số của phần tử trong mảng

`Tên_Mảng[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;`

`int N = A[1] + A[4];` $\rightarrow N = 12$

Ví dụ

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

0	2	4	6	8	10	12	14	16	18
A[0]	A[1]	A[2]	B[0]	B[1]	B[2]	B[3]	C[0]	C[1]	C[2]
i : 10									

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

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

$A[5] \Leftrightarrow B[2] \Leftrightarrow C[-2] \leftarrow$ nếu cấp phát liên tiếp

Mảng nhiều chiều

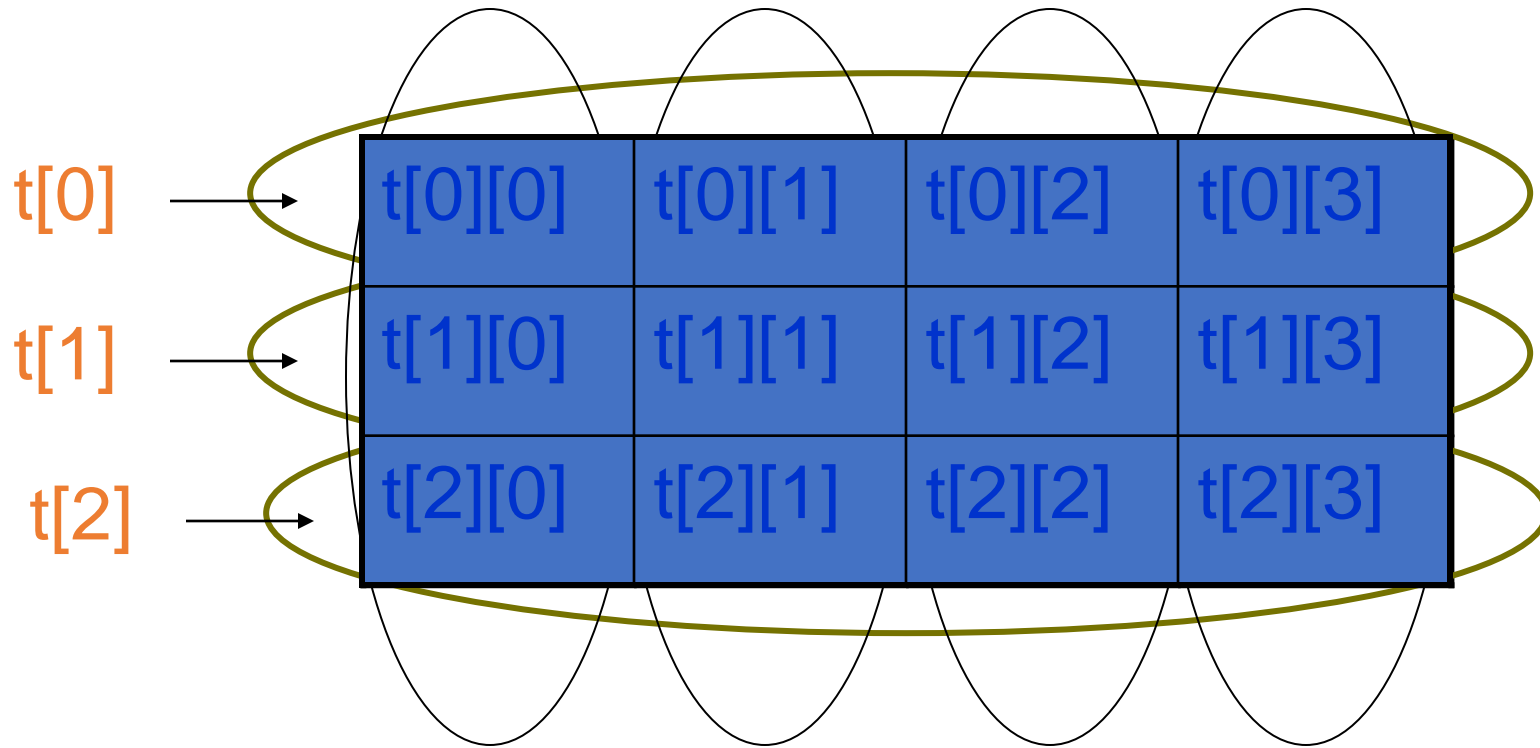
- Mỗi phần tử của mảng có thể là một mảng.

Kiểu Tên[Chiều_1] [Chiều_2]... [Chiều_N];

- Kiểu: 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**.

Mảng nhiều chiều

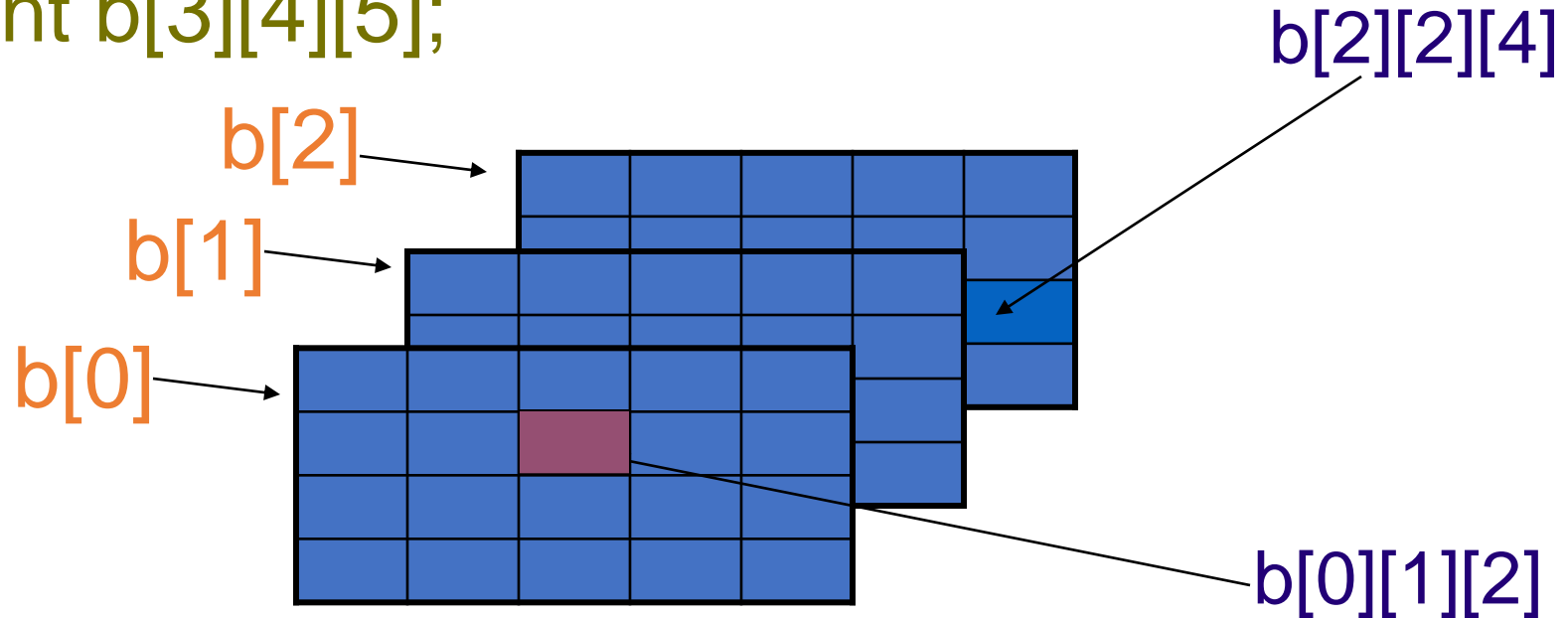
```
int t[3][4] ;
```



t[1][0] t[1][1] t[1][2] t[1][3]

Mảng nhiều chiều → Ví dụ

```
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 trống kích thước mảng.

```
int A1[8] = {2, 4, 6, 8, 10, 12, 14, 16};
```

```
int A2[] = {2, 4, 6, 8, 10, 12, 14, 16};
```

Các thao tác thường gặp

- Nhập/xuất dữ liệu cho mảng
 - Mảng 1 chiều, ma trận (mảng 2 chiều)
- 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()** nhập dữ liệu cho từng phần tử của mảng.

Ví dụ: int Table[10];

- Nhập dữ liệu cho một phần tử
`scanf("%d",&Table[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", &Table[i]);`
 - Nên in ra chỉ số phần tử khi nhập

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("Nhap luong mua thang %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]);
    }
}
```

Xuất dữ liệu từ mảng

Dùng hàm **printf()**

Ví dụ: `int Table[10];`

- Hiển thị phần tử thứ 5:

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

- Để hiển thị tất cả các phần tử:

```
for(i = 0; i < 10; i++)  
    printf("%4d", Table[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++)
    { //Nhập dữ liệu
        printf("A[%d]: ", i + 1);
        scanf("%d", &A[i]);
    }
    for (i = 0; i < MAX; i++)
        printf("%d\n", A[i]);
}
```


Kết quả

A[1]:	9	9	4	1	3
A[2]:	4	2	6	8	7
A[3]:	1	5	9	3	2
A[4]:	3				
A[5]:	2				
A[6]:	6				
A[7]:	8				
A[8]:	7				
A[9]:	5				
A[10]:	9				
A[11]:	3				
A[12]:	2				

9 4 1 3 2 6 8 7 5 9 3 2

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

```
#include <stdio.h>
int main() {
    int A[20][20], n, m, i, j;
    printf("Nhap so hang : "); scanf("%d", &n);
    printf("Nhap so cot : "); scanf("%d", &m);
    printf("\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++) {
            printf("Nhap phan tu A[%d,%d]: ", i + 1, j + 1);
            scanf("%d", &A[i][j]);
        }
    printf("\n\n MA TRAN DA NHAP \n\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++)
            printf("%4d", A[i][j]);
        printf("\n");
    }
}
```

Ví dụ 2→Kết quả thực hiện

Nhap so hang : 2

Nhap so cot : 4

Nhap phan tu $A[1,1]$: 12

Nhap phan tu $A[1,2]$: 34

Nhap phan tu $A[1,3]$: 54

Nhap phan tu $A[1,4]$: 3

Nhap phan tu $A[2,1]$: 123

Nhap phan tu $A[2,2]$: 872

Nhap phan tu $A[2,3]$: 12

Nhap phan tu $A[2,4]$: 34

MA TRAN DA NHAP

12	34	54	3
123	872	12	34

Đếm số phần tử thỏa mãn điều kiện

- Duyệt từng phần tử của dãy (*dùng **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("So thang mua nhieu hon 50mm: %d", dem);
```

Ví dụ: Nhập mảng, đưa ra TBC các số chia hết cho 7

```
#include <stdio.h>
int main() {
    int A[100];
    int n, i, d = 0, S = 0;
    printf("\n So phan tu cua mang (<100) : "); 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");
}
```

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
 - Nếu không đúng, so sánh tiếp với phần tử kế

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("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

- 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 của mảng có giá trị bằng k.

Tìm kiếm phần tử → Ví dụ

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

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

Dùng for

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

Dùng 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)

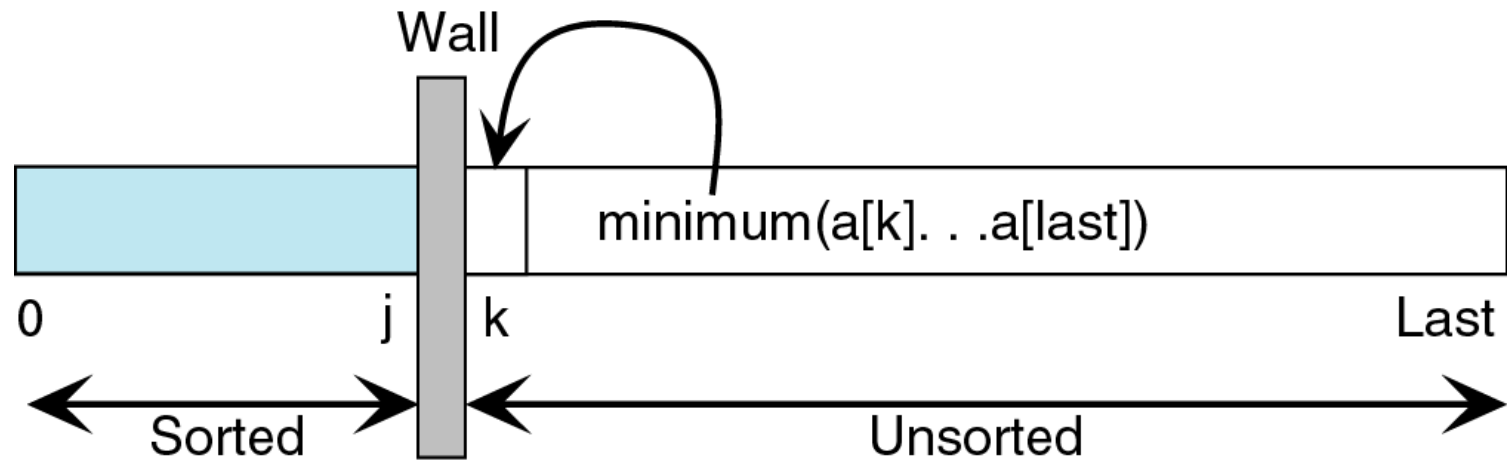
Bài toán sắp xếp tăng → Thuật toán lựa chọn

Nguyên tắc: Tại lượt sắp xếp 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

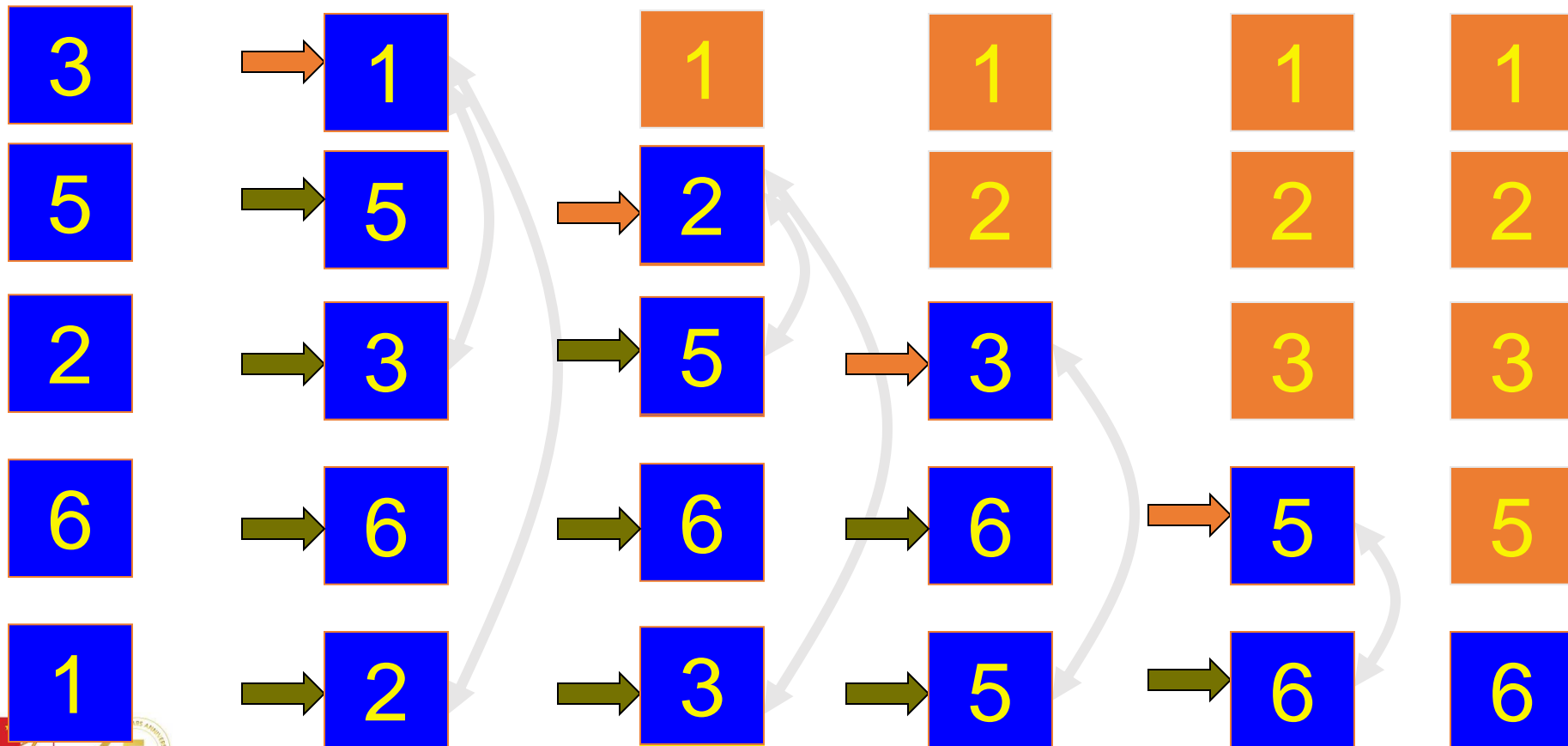
– Khi $k = 2$, phần tử thứ hai (chỉ số 1) đúng vị trí

– ...



Bài toán sắp xếp tăng → Thuật toán lựa chọn

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

Ví dụ

```
#include <stdio.h>
int main() {
    int A[100];
    int N, i, j, t;
    printf("So phan tu [< 100]: "); 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]);
}
```

Ví dụ

```
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;
        } //if & for_j
    printf("\nLuot %d : ", i + 1);
    for (j = 0; j < N; j++)
        printf("%4d", A[j]);
} //for_i
return 0;
} //main
```


Ví dụ → Kết quả

So phan tu [< 100]: 6

Hay nhap day so...

$A[1] = 5$

$A[2] = 32$

$A[3] = 67$

$A[4] = 12$

$A[5] = 24$

$A[6] = 16$

Day vua nhap...

5 32 67 12 24 16

Sap xep day theo thuat toan lua chon

Luot 1 : 67 5 32 12 24 16

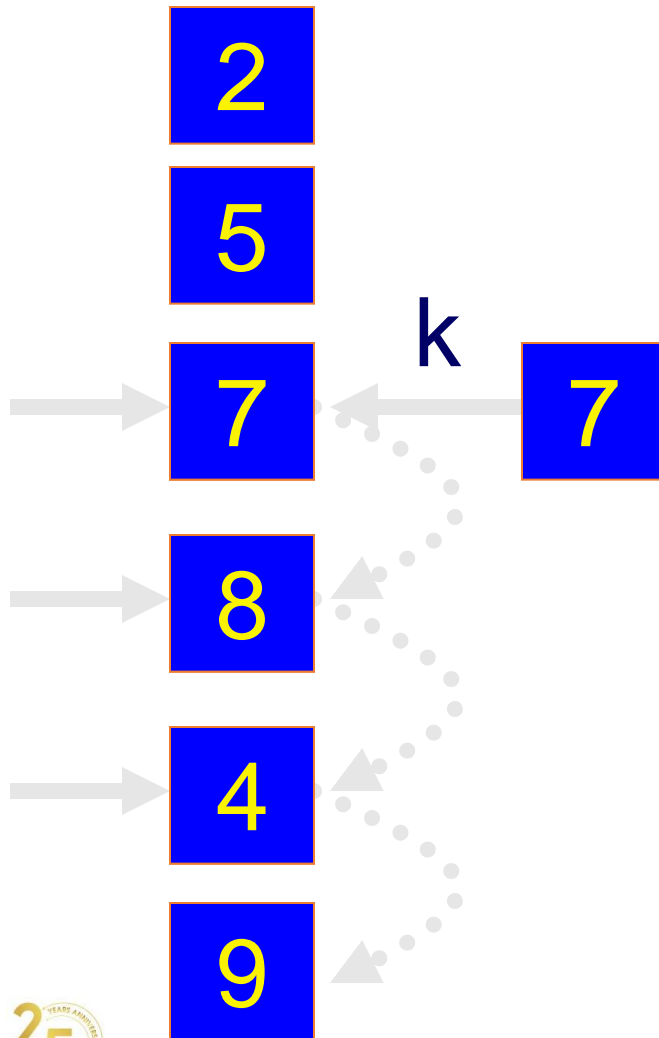
Luot 2 : 67 32 5 12 24 16

Luot 3 : 67 32 24 5 12 16

Luot 4 : 67 32 24 16 5 12

Luot 5 : 67 32 24 16 12 5

Bài toán chèn phần tử a vào vị trí k



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

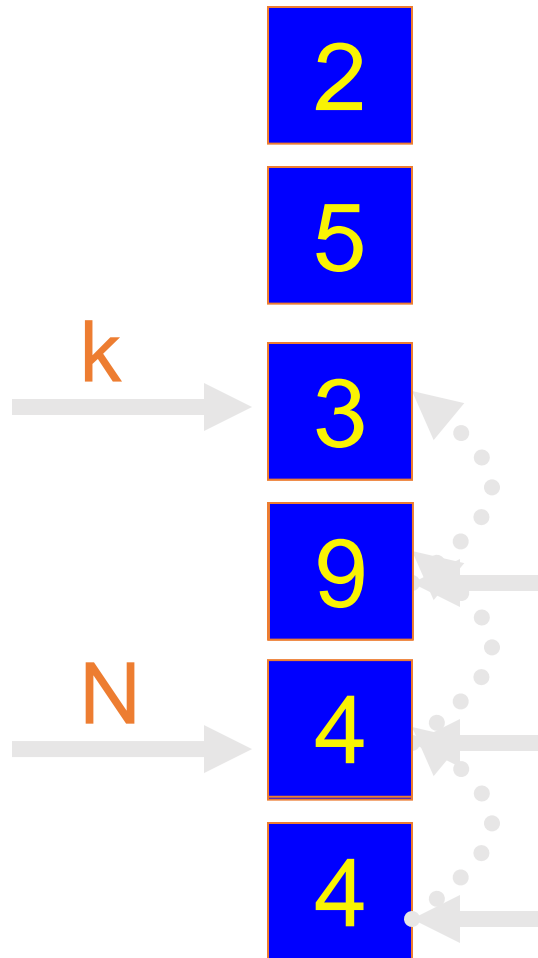
Chú ý:

$N = \text{MAX}$: không chèn được

$k > N \rightarrow$ Chèn vào vị trí N

$k < 0 \rightarrow$ 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;
```

Bài tập 1

1. Nhập vào dãy số, tính và đưa ra màn hình:
 - Tổng và tích của dãy số
 - Các số chia hết cho 3 và lớn hơn 10
 - Đếm các số nằm trong đoạn [100,1000]
2. Nhập vào một dãy số, tìm số chẵn nhỏ nhất.
3. Nhập dãy số, đếm xem có bao nhiêu bộ 3 số thỏa mãn điều kiện $x_i = (x_{i-1} + x_{i+1})/2$
4. Nhập vào một dãy số, đưa ra số bé nhất và vị trí những số bằng số bé nhất.
5. Nhập vào n và 2 dãy số (x_1, x_2, \dots, x_n) , (y_1, y_2, \dots, y_n) rồi tính:

$$a) \sum_{i=1}^n \cos x_i \sin y_i \quad b) \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Bài tập 2

1. Nhập vào một dãy số nguyên. 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. Nhập vào dãy số có n phần tử. Đọc số x và số k nguyên. Chèn x vào vị trí k của dãy. Nếu $k > n$, chèn x vào vị trí $n+1$.
3. Nhập vào một dãy số 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, xóa đi các phần tử chia hết cho 5 và đưa kết quả ra màn hình.

Bài chữa

```
#include <stdio.h>
int main() {
    int A[100];
    int N, i;
    // Nhập dữ liệu
    printf("Số phần tử : ");
    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        printf("A[%d] = ", i);
        scanf("%d", &A[i]);
    }
    // Các thao tác xử lý mảng: chèn, xóa, sắp xếp,...
    // Đưa dữ liệu ra
    for (i = 0; i < N; i++)
        printf("%4d", A[i]);
}
```

Bài chữa → Sắp xếp số chẵn chia hết 3 lên đầu dãy ...

```
{
    int d = 0, t;
    for (i = 0; i < N; i++) // Xếp số chẵn chia hết cho 3 lên đầu dãy
        if (A[i] % 6 == 0)
        {
            t = A[i];
            A[i] = A[d];
            A[d] = t;
            d++;
        }
    for (i = d; i < N; i++) // Sau đó là các số không chia hết cho 3
        if (A[i] % 3 != 0)
        {
            t = A[i];
            A[i] = A[d];
            A[d] = t;
            d++;
        }
}
```

Bài chữa → Sắp xếp số chẵn chia hết 3 lên đầu dãy ... → Kết quả

Số phần tử : 9

$A[0] = 1$

$A[1] = 2$

$A[2] = 6$

$A[3] = 4$

$A[4] = 9$

$A[5] = 7$

$A[6] = 8$

$A[7] = 3$

$A[8] = 5$

6 2 1 4 7 8 5 3 9

Bài chữa → Sắp xếp tăng dần và chèn đúng vị trí

```
{
    int k = 0, i, j, t;
    for (i = 0; i < N - 1; i++) //Sắp xếp lựa chọn
        for (j = i + 1; j < N; j++)
            if (A[i] > A[j]) {
                t = A[j];
                A[j] = A[i];
                A[i] = t;
            }
    printf("Phan tu moi:"); scanf("%d", &k); //Nhập p/tử mới
    i = N; //Chèn đúng vị trí
    while ((i > 0) && (A[i - 1] > k)) {
        A[i] = A[i - 1];
        i--;
    }
    A[i] = k;
    N++;
}
```

Bài chữa → Sắp xếp tăng dần và chèn đúng vị trí → Kết quả

So phan tu : 6

$A[0] = 7$

$A[1] = 2$

$A[2] = 4$

$A[3] = 5$

$A[4] = 3$

$A[5] = 1$

Phan tu moi:6

1 2 3 4 5 6 7

Bài chữa → Xóa các phần tử chia hết cho 5

```
{  
    // PP: Giữ lại các phần tử không chia hết cho 5  
    int d = 0, i;  
    for (i = 0; i < N; i++)  
        if (A[i] % 5 != 0)  
        {  
            A[d] = A[i];  
            d++;  
        }  
    N = d;  
}
```

Bài chữa → Xóa các phần tử chia hết cho 5 → Kết quả

So phan tu : 10

A[0] = 5

A[1] = 1

A[2] = 21

A[3] = 15

A[4] = 10

A[5] = 34

A[6] = 23

A[7] = 45

A[8] = 54

A[9] = 32

1 21 34 23 54 32

Bài tập 3: 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
- 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

$$s_{i,j} = \begin{cases} 1 & \text{nếu } u_{i,j} > 0 \\ 0 & \text{nếu } u_{i,j} = 0 \\ -1 & \text{nếu } u_{i,j} < 0 \end{cases}$$

Nhập vào một ma trận vuông,..

```
#include <stdio.h>
int main()
{
    int A[20][20], N, i, j;
    printf("Nhap kích thước : "); scanf("%d", &N);
    printf("\n");
    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++) {
            printf("Nhap phần tử [%d,%d]:", i + 1, j + 1);
            scanf("%d", &A[i][j]);
        }
    printf("\n\n MA TRẬN ĐÃ NHẬP \n\n");
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++)
            printf("%4d", A[i][j]);
        printf("\n");
    }
}
```

Đưa ra ma trận tam giác trên, dưới

```
printf("\n\n MA TRAN TAM GIAC TREN \n\n");
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++)
        if (j >= i)
            printf("%4d", A[i][j]);
        else
            printf("%4c", 32); //32 là mã ASCII của dấu cách
    printf("\n");
}
printf("\n\n MA TRAN TAM GIAC DUOI \n\n");
for (i = 0; i < N; i++) {
    for (j = 0; j <= i; j++)
        printf("%4d", A[i][j]);
    printf("\n");
}
return 0;
} //main
```

Kết quả

Nhap kích thước : 4

Nhap phan tu [1,1]:12

Nhap phan tu [1,2]:34

Nhap phan tu [1,3]:52

Nhap phan tu [1,4]:9

Nhap phan tu [2,1]:10

Nhap phan tu [2,2]:52

Nhap phan tu [2,3]:54

Nhap phan tu [2,4]:75

Nhap phan tu [3,1]:8

Nhap phan tu [3,2]:12

Nhap phan tu [3,3]:45

Nhap phan tu [3,4]:7

Nhap phan tu [4,1]:11

Nhap phan tu [4,2]:42

Nhap phan tu [4,3]:22

Nhap phan tu [4,4]:34

MA TRAN DA NHAP

12 34 52 9

10 52 54 75

8 12 45 7

11 42 22 34

MA TRAN TAM GIAC TREN

12 34 52 9

52 54 75

45 7

34

MA TRAN TAM GIAC DUOI

12

10 52

8 12 45

11 42 22 34

Chương 5: Mảng, con trỏ và chuỗi ký tự

5.1. Mảng

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

5.2. Con trỏ

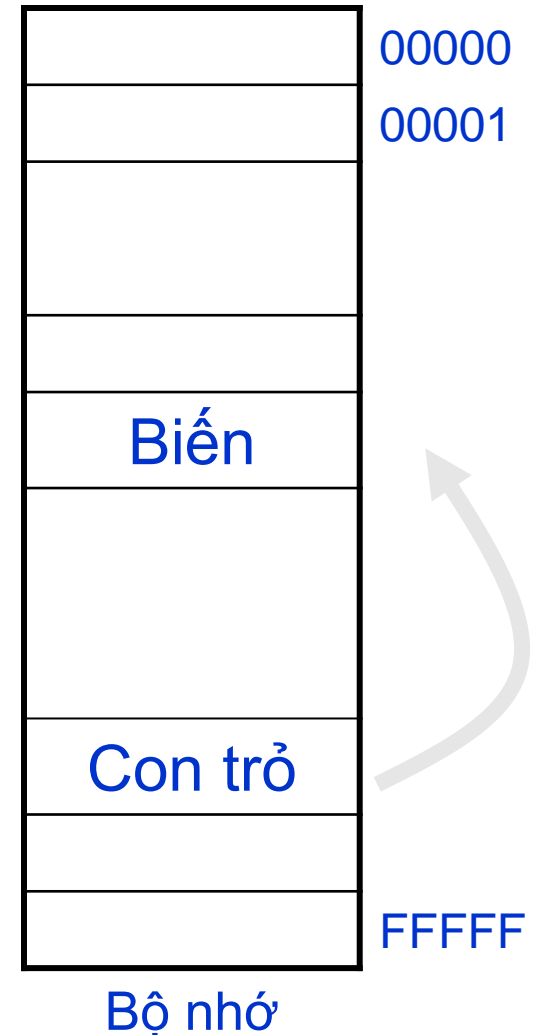
- Khái niệm và cách khai báo
- Toán tử địa chỉ (&), toán tử nội dung (*)
- Phép toán trên con trỏ
- Con trỏ và mảng

5.3. Chuỗi ký tự

- Khái niệm, khai báo và sử dụng
- Các hàm xử lý ký tự và chuỗi ký tự
- Mảng chuỗi 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 lưu trữ một byte dữ liệu
 - 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 phát 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

```
int N;  
float x;  
char Arr[4];
```

```
N=1000; // 03E8
```

```
X=9.6875; // 411B0000
```

```
for(i=0; i<4; i++)
```

```
    Arr[i]=4*i+1;
```

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

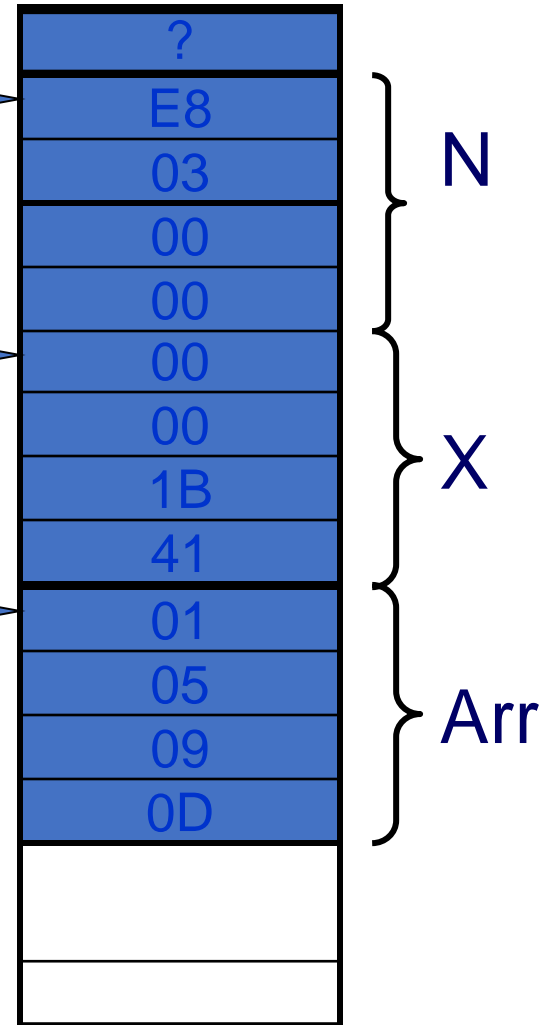
Địa chỉ biến N



Địa chỉ biến X



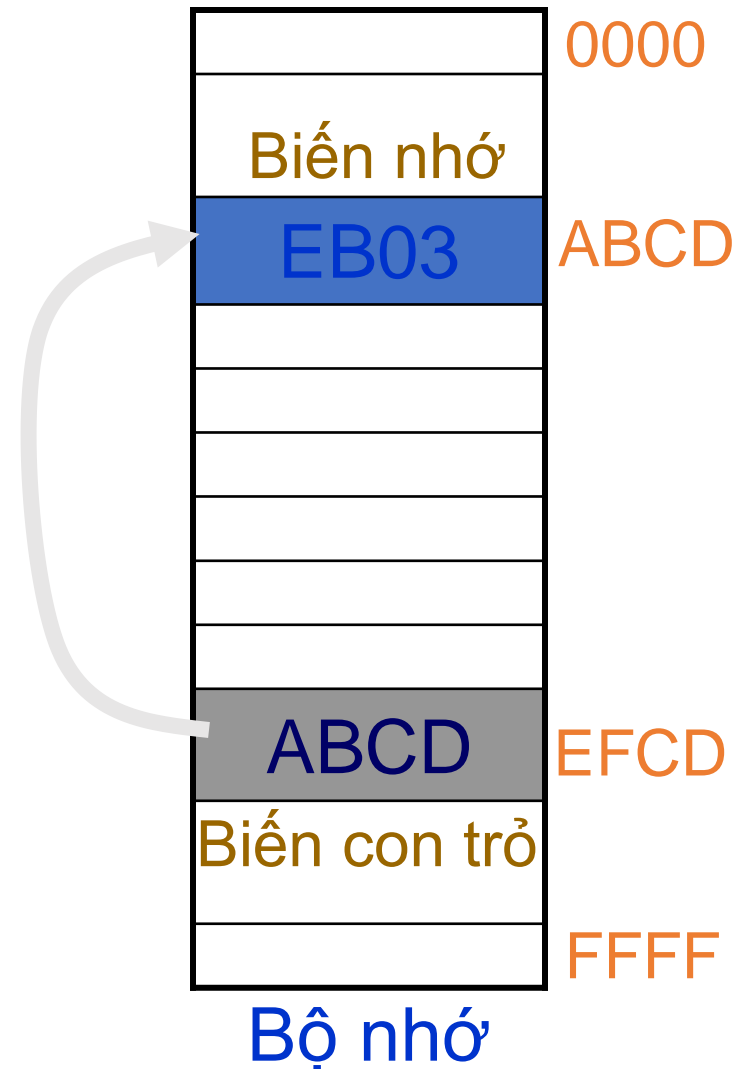
Địa chỉ biến Arr



Bộ nhớ

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, kí 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 trỏ → Khai báo

Kiểu * Tên;

- Tên: Tên của một biến con trỏ
- Kiểu: Kiểu của biến mà con trỏ “Tên” trỏ tới
 - Giá trị của con trỏ có thể thay đổi được
 - Trỏ tới các biến khác nhau, có cùng kiểu
 - Kiểu biến mà con trỏ trỏ tới không thay đổi được
 - Muốn thay đổi phải thực hiện “ép kiểu”
- Ví dụ:

```
int * pi; // Con trỏ, trỏ tới một biến kiểu nguyên  
char * pc; // Con trỏ, trỏ tới một biến kiểu kí tự
```

Toán tử địa chỉ (&)

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

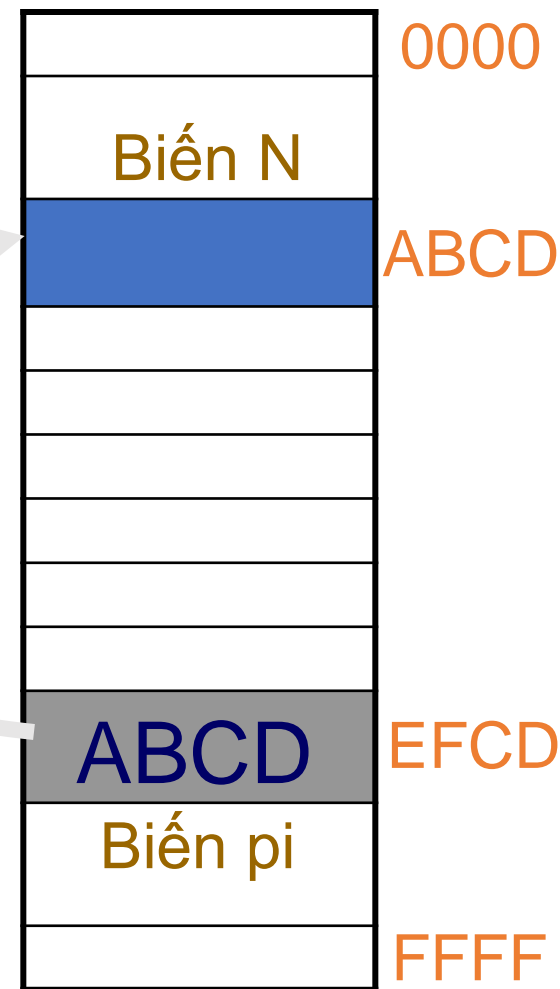
Địa chỉ biến có thể được gán cho một con trỏ, trỏ tới đối tượng cùng kiểu.

- Ví dụ

```
short int N; // &N → ABCD
```

```
short int *pi;
```

```
pi = &N; // pi ← ABCD
```



Bộ nhớ

Toán tử nội dung (*)

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

• Ví dụ

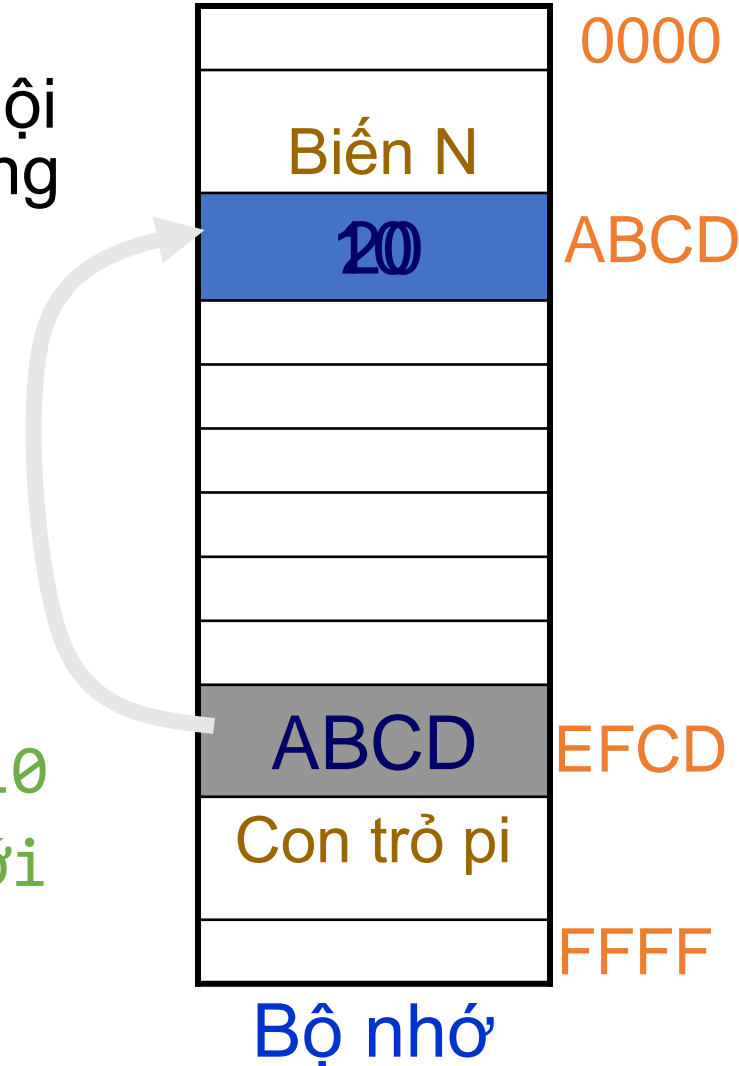
```
short int N;
```

```
short int * pi;
```

```
pi = &N;
```

```
N = 10; // Vùng nhớ mà pi trỏ  
tới mang giá trị 10; Vậy *pi=10
```

```
*pi = 20; // Vùng nhớ pi trỏ tới  
được gán giá trị 20; Vậy N=20
```



Gán giá trị cho con trỏ

- Con trỏ được gán địa chỉ của một biến
 - Biến cùng kiểu với kiểu mà con trỏ trỏ tới
Nếu không, cần phải ép kiểu
- Con trỏ được gán giá trị của con trỏ khác
 - Hai con trỏ sẽ trỏ tới cùng một biến (do cùng địa chỉ)
 - Hai con trỏ nên cùng kiểu trỏ 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 trỏ tới.
Ví dụ: `int *p1, *p2; *p1 = *p2;`

Ví dụ

```
#include <stdio.h>
int main()
{
    int N = 5, M = 10;
    int *p1 = &N;
    int *p2 = &M;
    *p1 = *p2;
    printf("%d %d", *p1, *p2);
}
```

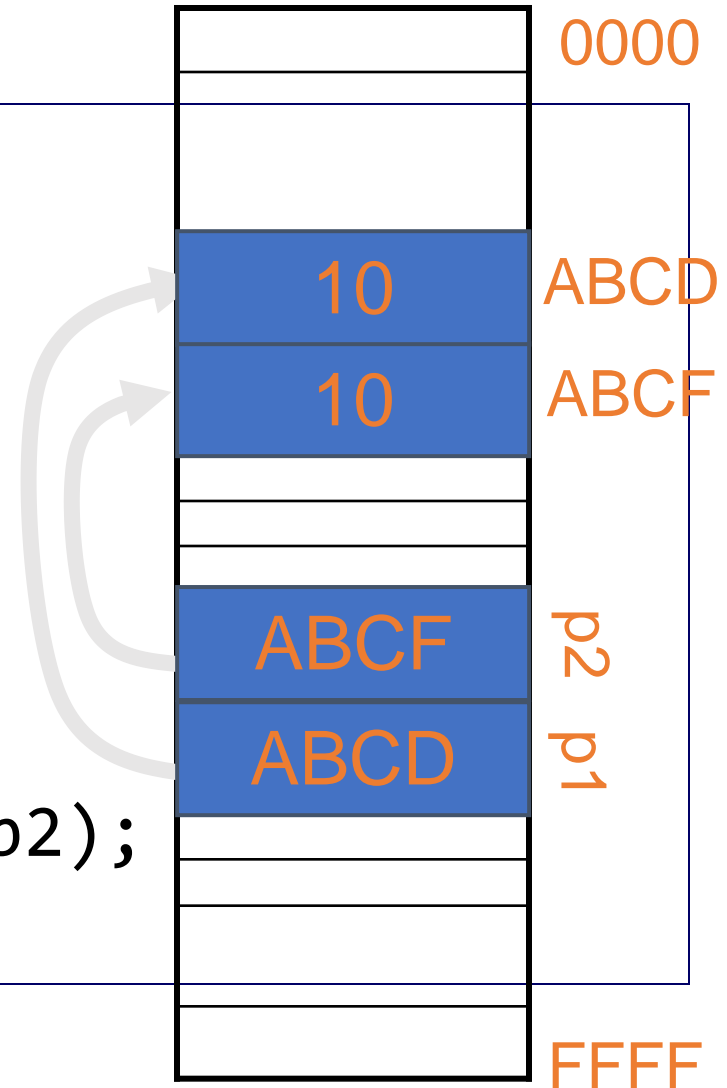
10 10

```
#include <stdio.h>
int main()
{
    int N = 5, M = 10;
    int *p1 = &N;
    int *p2 = &M;
    p1 = p2;
    printf("%d %d", *p1, *p2);
}
```

10 10

Ví dụ → Trường hợp 1

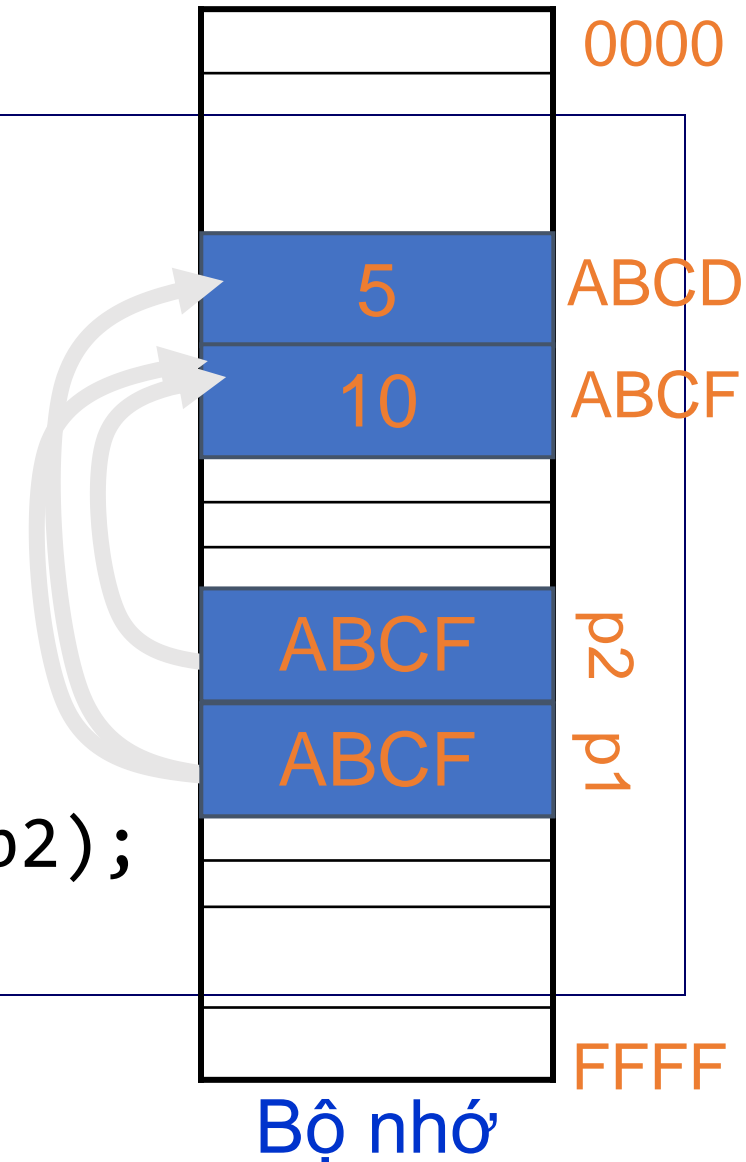
```
#include <stdio.h>
int main()
{
    int N = 5, M = 10;
    int *p1 = &N;
    int *p2 = &M;
    *p1 = *p2;
    printf("%d %d", *p1, *p2);
}
```



Bộ nhớ

Ví dụ → Trường hợp 2

```
#include <stdio.h>
int main()
{
    int N = 5, M = 10;
    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

```
short int N=1000,M=2000,P=3000;  
short int *p1=&P, *p2=&N;
```

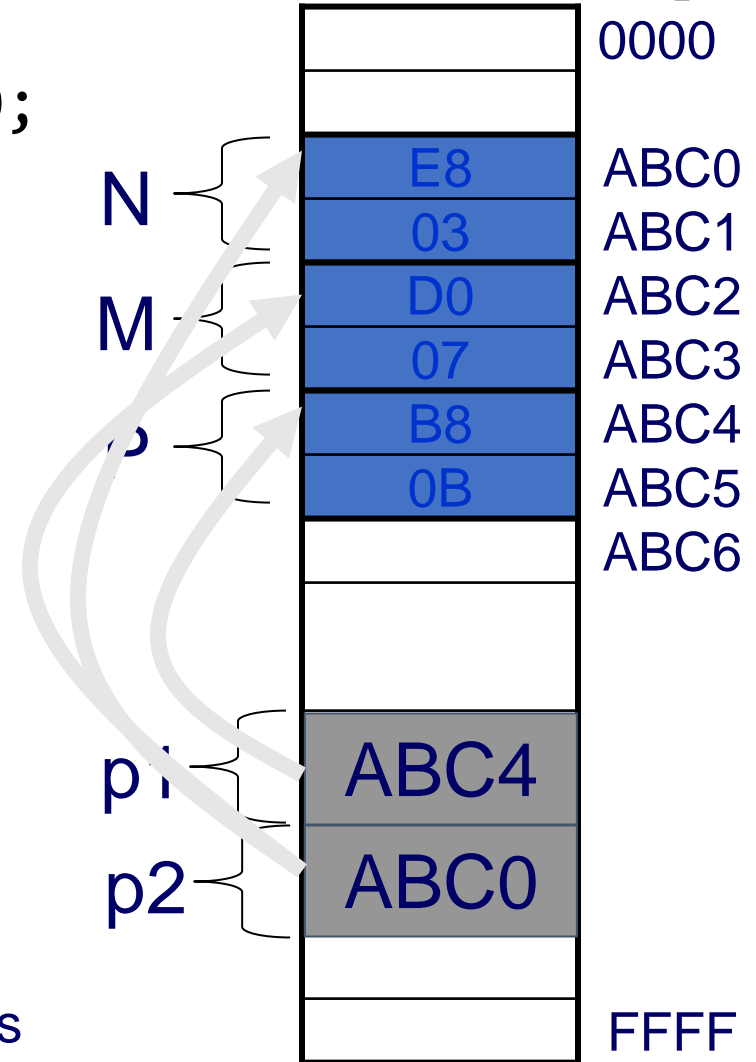
$p2 - p1 \rightarrow -2$

$* (p1-1) \rightarrow 2000$

$* ++p2 \rightarrow 2000$

Ghi chú:

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



Bộ nhớ

Mối 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 trỏ void

```
void * Tên_con_trỏ;
```

- Là một con trỏ đặc biệt: con trỏ 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 arr[2][2][2] = {10, 2, 3, 4, 5, 6, 7, 8};
    int *p, *q;
    p = &arr[1][1][1];
    q = (int *) arr;
    printf("%d, %d\n", *p, *(q+4));
    return 0;
}
```

8, 5

Chương 5: Mảng, con trỏ và chuỗi ký tự

5.1. Mảng

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

5.2. 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

5.3. Chuỗi ký tự

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

Khái niệm chuỗi ký tự

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

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

Khái niệm chuỗi ký tự → Lưu ý

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

Ví dụ

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

```
char str[8] = "Tin học";
```

Truy nhập phần tử của xâu

Giống như truy nhập tới một phần tử của mảng kí tự

tên_xâu [chỉ_số_của_kí_tự]

Ví dụ: char Str[10] = "Tin hoc";

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

Str[0] → 'T'

Str[3] = '-';

Str[3] → ' '

Str[7] = ' ';

Str[7] → '\0'

Str[8] = '1' ;

Str[8] → ?

Str[9] = '\0';

Str: Tin-hoc 1

Ví dụ: Nhập chuỗi và đếm số kí tự '*'

```
#include <stdio.h>
void main()
{
    char Str[100];
    int d = 0, i = 0;
    printf("Nhap xau ki tu: "); gets(Str);
    while (Str[i] != '\0') {
        if (Str[i] == '*') d++;
        i++;
    }
    printf("Ket qua : %d", d);
}
```

Ví dụ: Nhập chuỗi và đếm số kí tự '*'

Nhap xau ki tu: ** abc*** dd*e**e*dd

Ket qua : 9

Nhap xau ki tu: ***** ** * * *

Ket qua : 13

Ví dụ: Nhập câu và đưa ra dưới dạng cột

Nhap xau: Dai Hoc Bach Khoa

Dai

Hoc

Bach

Khoa

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

Ví dụ: Nhập câu và đưa ra dưới dạng cột

```
#include <stdio.h>
void main()
{
    char S[100];
    int i = 0;
    printf("Nhap xau: "); gets(S);
    while (S[i] != '\0') {
        if (S[i] != 32 && S[i + 1] == 32)
            printf("%c\n", S[i]);
        else if (S[i] != 32)
            printf("%c", S[i]);
        i++;
    }
    printf("\n\n");
}
```

Các hàm xử lý kí tự

Tập tiêu đề : `ctype.h`

```
#include <ctype.h>
```

Các hàm xử lý kí 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 tục <C/K>?: "); fflush(stdin);  
} while(toupper(getche()) != 'K');
```

Các hàm xử lý kí 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'));` => 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'));` => ≠ 0

Các hàm xử lý kí 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`
- `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 kí tự → Kiểm tra kí tự đặc biệt

- `int iscntrl(char ch);`
 - Kiểm tra kí 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 kí 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>
void main()
{
    int i = 0;
    char S[50];
    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);
}
```

Nhập một xâu: Study, study more, ...

Xâu ban đầu: Study, study more,

Xâu kết quả: STUDY, STUDY MORE, ...

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

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

Ví dụ: Nhập sâu và đếm từ, phân cách bởi dấu trắng

Nhap xau ky tu: Hello World

Ket qua : 2

Nhap xau ky tu: Hello

Ket qua : 1

Nhap xau ky tu: Hello

Ket qua : 1

Các hàm xử lý chuỗi ký tự

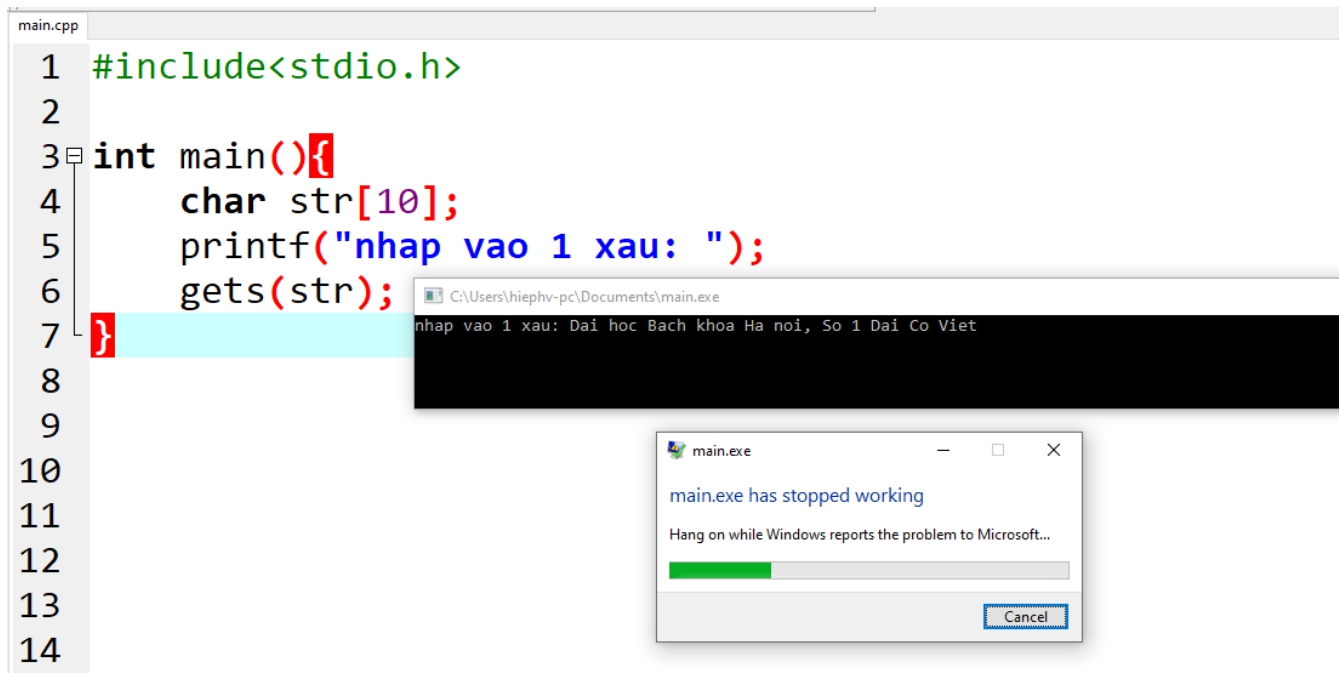
Vào/ra chuỗi ký tự

- Tập tiêu đề: `stdio.h`
- Nhập chuỗi ký tự
 - `gets(tên_xâu);`
 - `scanf("%s", &tên_xâu);`
- Hiển thị chuỗi ký tự
 - `puts(tên_xâu);`
 - `printf("%s", tên_xâu);`

Sự khác nhau giữa **gets** và **scanf**?

Lưu ý:

- Hàm **gets**: các trình biên dịch mới hiện tại không còn hỗ trợ nữa
 - Lý do: gets **không kiểm tra kích thước xâu được nhập**



The screenshot shows a code editor with a file named `main.cpp`. The code is as follows:

```
1 #include<stdio.h>
2
3 int main(){
4     char str[10];
5     printf("nhap vao 1 xau: ");
6     gets(str);
7 }
8
9
10
11
12
13
14
```

Below the code editor, a console window shows the output: `nhap vao 1 xau: Dai hoc Bach khoa Ha noi, So 1 Dai Co Viet`. This input is clearly longer than the `10` characters allocated for the `str` array, demonstrating a buffer overflow.

Overlaid on the bottom right is a Windows error dialog box titled `main.exe` with the message: `main.exe has stopped working`. It includes a progress bar and a `Cancel` button, indicating that the program crashed due to the memory error.

- Giải pháp: dùng `fgets(str, so_ky_tu_toida, stdin);`

Các hàm xử lý chuỗi ký tự

Tập 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 chuỗi 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 chuỗi ký tự

```
int strcmp(const char *xâu_1, const char *xâu_2);
```

- So sánh hai chuỗi.
- Trả về giá trị 0 nếu hai chuỗi 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 chuỗi ký tự

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

- Ghép nối chuỗi nguồn vào ngay sau chuỗi đích, trả lại chuỗi 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 chuỗi 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 chuỗi: `strcpy`, `strcat` cũng đã không dùng được đối với các trình biên dịch mới.
- Tuy nhiên trong phạm vi môn Tin đại cương, chúng ta vẫn chấp nhận dùng các hàm trên với trình biên dịch phù hợp.

Các hàm xử lý chuỗi ký tự (tiếp)

Tập tiêu đề: `stdlib.h`

- `int atoi(const char * str);`
 - Chuyển một chuỗi ký tự thành một số nguyên tương ứng
 - Ví dụ: `atoi("1234") → 1234`
- `int atol(const char * str);`
 - Chuyển chuỗi ký tự thành số long int
- `float atof(const char * str);`
 - Chuyển chuỗi ký tự thành số thực
 - Ví dụ: `atof("123.456E-2") → 1.23456`
- Nếu thất bại, hàm (cả 3 hàm) trả về 0

Các hàm xử lý chuỗi 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 chuỗi 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);
}
```

a = 10, b = 20, c = 30

Các hàm xử lý chuỗi 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 chuỗi ký tự**.

```
#include <stdio.h>
int main()
{
    char str[32];
    int a = 10;
    float b = 20;
    char str2[10] = "abc";
    sprintf(str, "%d %f %s", a, b, str2);
    printf("str = %s", str);
}
```

str = 10 20.000000 abc

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

```
#include <stdio.h>
#include <string.h>
void main()
{
    int d = 0;
    char S1[50], S2[20], *p;
    printf("Nhap xau thu nhat: "); gets(S1);
    printf("Nhap xau thu hai: "); gets(S2);
    p = strstr(S1, S2);
    while (p != NULL)
    {
        d = d + 1;
        p = strstr(p + 1, S2); //vi tri timkiem ke tiep
    }
    printf("Xau \"%s\" x/hien trong xau \"%s\" %d lan", S2, S1, d);
}
```


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

Nhap xau thu nhat: abcabcabca

Nhap xau thu hai: bc

Xau "bc" x/hien trong xau "abcabcabca" 3 lan

Nhap xau thu nhat: aaaaaa

Nhap xau thu hai: aa

Xau "aa" x/hien trong xau "aaaaaa" 4 lan

Ví dụ 2: Kiểm tra chuỗi đối xứng

```
#include <stdio.h>
#include <string.h>
void main()
{
    char s[20];
    int i, n;
    printf("Nhap vao xau ki tu: "); gets(s);
    n = strlen(s);
    for (i = 0; i < n / 2; i++)
        if (s[i] != s[n - 1 - i])
            break;
    if (i == n / 2)
        printf("Xau doi xung");
    else
        printf("Xau khong doi xung");
}
```

Ví dụ 3: Đảo ngược chuỗi ký tự

```
#include <stdio.h>
#include <string.h>
main()
{
    char s[100], c;
    int i, n;
    printf("Nhap xau: "); gets(s);
    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("%s", s);
}
```

Ví dụ 4: Đếm số lần xuất hiện chữ cái trong xâu

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
void main()
{
    char s[64];
    int dem[26] = {};
    int i, n;
    puts("Nhap vao xau 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("Ki tu %c xuat hien %d lan\n", 'a' + i, dem[i]);
}
```

Ví dụ 4: Đếm số lần xuất hiện chữ cái trong xâu

Nhap vao xau ki tu:

Study, study more, study forever

Ki tu d xuất hiện 3 lần

Ki tu e xuất hiện 3 lần

Ki tu f xuất hiện 1 lần

Ki tu m xuất hiện 1 lần

Ki tu o xuất hiện 2 lần

Ki tu r xuất hiện 3 lần

Ki tu s xuất hiện 3 lần

Ki tu t xuất hiện 3 lần

Ki tu u xuất hiện 3 lần

Ki tu v xuất hiện 1 lần

Ki tu y xuất hiện 3 lần

Mảng chuỗi ký tự

- Chuỗi 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 chuỗi có độ dài tối đa 30

- Sử dụng

- Như một mảng bình thường
- Mỗi phần tử mảng được sử dụng như một chuỗi ký tự

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

```
#include <stdio.h>
void main()
{
    int i, n;
    char DS[100][30];
    printf("Nhap DSSV (<100), go Enter de thoat..\n");
    n = 0;
    do {
        printf("Ten sinh vien[%d]: ", n + 1); gets(DS[n]);
        if (DS[n][0] != '\0') n++;
        else break;
        if (n == 100) break;
    } while (1);
    printf("\n\nDS sinh vien vua nhap \n");
    for (i = 0; i < n; i++)
        printf("%s\n", DS[i]);
}
```

Ví dụ → Kết quả thực hiện

Nhap DSSV (<100), go Enter de thoat..

Ten sinh vien[1]: Pham Viet Linh

Ten sinh vien[2]: Ha Minh Duc

Ten sinh vien[3]: Nguyen Binh An

Ten sinh vien[4]: Bui Thanh Trung

Ten sinh vien[5]: Tran Thanh Tung

Ten sinh vien[6]: Vu Tu Anh

Ten sinh vien[7]: Phung Nhat Huy

Ten sinh vien[8]: Le Quang Nam

Ten sinh vien[9]:

DS sinh vien vua nhap

Pham Viet Linh

Ha Minh Duc

Nguyen Binh An

Bui Thanh Trung

Tran Thanh Tung

Vu Tu Anh

Phung Nhat Huy

Le Quang Nam

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

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

Nhập dãy (<100) xâu cho tới khi gặp xâu “***”. Đưa ra màn hình xâu có độ dài lớn nhất

Nhap xau thu [1]: Paris
Nhap xau thu [2]: Rome
Nhap xau thu [3]: Luxembourg
Nhap xau thu [4]: Santiago
Nhap xau thu [5]: Moscow
Nhap xau thu [6]: New Delhi
Nhap xau thu [7]: Washington
Nhap xau thu [8]: Madrid
Nhap xau thu [9]: ***

Xau dai nhat la: Luxembourg, co do dai :10

Nhap xau thu [1]: a
Nhap xau thu [2]: ab
Nhap xau thu [3]:
Nhap xau thu [4]: b
Nhap xau thu [5]: ***

Xau dai nhat la: ab, co do dai :2

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

```
#include <stdio.h>
#include <string.h>
void main()
{
    int i, j, N;
    char DS[100][30], str[30];
    //Nhập DS sinh viên
    printf("Số sinh viên : ");
    scanf("%d", &N);
    fflush(stdin);
    for (i = 0; i < N; i++)
    {
        printf("Tên sinh viên[%d]: ", i);
        gets(DS[i]);
    }
}
```

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

```
//So sánh theo Họ+đệm+tên
for (i = 0; i < N - 1; i++)
    for (j = i + 1; j < N; j++)
        if (strcmp(DS[i], DS[j]) > 0)
        {
            strcpy(str, DS[i]);
            strcpy(DS[i], DS[j]);
            strcpy(DS[j], str);
        }
//In danh sách đã sắp xếp
printf("\nDS sinh vien vua nhap \n");
for (i = 0; i < N; i++)
    printf("%s\n", DS[i]);
} //main
```

Ví dụ → Kết quả thực hiện

Số sinh viên : 10
Ten sinh viên[0]: Nguyen Hoai Nam
Ten sinh viên[1]: Le Tu Anh
Ten sinh viên[2]: Tran Thai Ha
Ten sinh viên[3]: Nguyen Van Tuan
Ten sinh viên[4]: Tran Thanh Son
Ten sinh viên[5]: Tran Hoai Nam
Ten sinh viên[6]: Bui Truong Son
Ten sinh viên[7]: Le Thanh Tuan
Ten sinh viên[8]: Le Thanh Son
Ten sinh viên[9]: Nguyen Nam Giang

DS sinh viên vừa nhập
Bui Truong Son
Le Thanh Son
Le Thanh Tuan
Le Tu Anh
Nguyen Hoai Nam
Nguyen Nam Giang
Nguyen Van Tuan
Tran Hoai Nam
Tran Thai Ha
Tran Thanh Son

Ví dụ : Sắp xếp theo tên

```
//Sap xep theo ten
char ten_i[30], ten_j[30];
for (i = 0; i < N - 1; i++)
    for (j = i + 1; j < N; j++)
    {
        strcpy(ten_i, strrchr(DS[i], 32)); //trích ra từ cuối
        strcpy(ten_j, strrchr(DS[j], 32)); //của xâu họ&tên
        if (strcmp(ten_i, ten_j) > 0)
        {
            strcpy(str, DS[i]);
            strcpy(DS[i], DS[j]);
            strcpy(DS[j], str);
        }
    }
}
```

Ví dụ → Kết quả thực hiện

So sinh vien : 10
Ten sinh vien[0]: Nguyen Hoai Nam
Ten sinh vien[1]: Le Tu Anh
Ten sinh vien[2]: Tran Thai Ha
Ten sinh vien[3]: Nguyen Van Tuan
Ten sinh vien[4]: Tran Thanh Son
Ten sinh vien[5]: Tran Hoai Nam
Ten sinh vien[6]: Bui Truong Son
Ten sinh vien[7]: Le Thanh Tuan
Ten sinh vien[8]: Le Thanh Son
Ten sinh vien[9]: Nguyen Nam Giang

DS sinh vien vua nhap
Le Tu Anh
Nguyen Nam Giang
Tran Thai Ha
Tran Hoai Nam
Nguyen Hoai Nam
Bui Truong Son
Le Thanh Son
Tran Thanh Son
Nguyen Van Tuan
Le Thanh Tuan

Bài tập

1. Nhập vào xâu kí tự, đếm số kí tự chữ hoa, chữ thường và chữ số trong xâu.

2. Nhập xâu kí tự, chuyển các chữ cái đầu các từ thành chữ hoa, các chữ cái còn lại thành chữ thường.

Ví dụ: người sử dụng nhập vào

“Xin cHao caC bAn”

Kết quả in ra là:

“Xin Chao Cac Ban”

3. Nhập xâu kí tự, xóa các dấu cách dư thừa trong xâu.

Ví dụ: người sử dụng nhập vào

“ Xin chao cac ban ”

Chương trình sẽ in ra

“Xin chao cac ban”

Bài tập

4. Nhập vào 2 chuỗi S1, S2 và một số nguyên k. Hãy chèn chuỗi S1 vào S2 tại vị trí k và đưa ra màn hình (*giả thiết chuỗi S2 được khai báo đủ lớn*).
5. Một văn bản gồm không quá 60 dòng, mỗi dòng không quá 80 kí tự. Hãy viết chương trình thực hiện nhập vào một văn bản, sau đó
 - a. Nhập vào chuỗi s và chỉ ra vị trí xuất hiện của chuỗi 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