

CHƯƠNG 6

Con trỏ và mảng

Nội Dung

- Con trỏ và địa chỉ
- Tham số con trỏ
- Con trỏ và mảng
- Số học trên con trỏ
- Chuỗi – con trỏ đến ký tự
- Mảng con trỏ - con trỏ đến con trỏ
- Mảng nhiều chiều
- So sánh mảng con trỏ và mảng nhiều chiều
- Khởi động mảng
- Tham số dòng lệnh

Con trỏ và địa chỉ

- Con trỏ là đại lượng biểu diễn địa chỉ của một đối tượng khác. Ta dùng biến con trỏ để chứa địa chỉ của một biến đã có.
- Con trỏ được dùng nhiều trong ngôn ngữ C.
- Con trỏ và mảng liên hệ chặt chẽ với nhau.

Con trỏ và địa chỉ

- Bộ nhớ có thể được hình như một bảng gồm các ô nhớ được đánh số để chứa dữ liệu. Mỗi ô nhớ là một byte (8 bit) dữ liệu.
- Số được đánh tương ứng với vị trí của ô nhớ tương ứng là địa chỉ của ô nhớ đó.
- Một biến trong C khi định nghĩa sẽ được lưu trữ trong bộ nhớ và có địa chỉ tương ứng.

Con trỏ và địa chỉ

```
void main()
{
    int m;
    int n = 40;
    // ...
}
```

Symbol	Addr	Value
	00000000	
	00000001	
	.	
	.	
n	003DFB0C	40
	.	
	.	
	.	
m	003DFB18	???
	.	
	.	
	.	

Con trỏ

- Ta dùng biến con trỏ để chứa địa chỉ của một biến đã có.
- Phép toán một ngôi & trả về địa chỉ của một đối tượng.
- Phép toán một ngôi * trả về nội dung của đối tượng mà một biến con trỏ đang trỏ đến.

Con trỏ và địa chỉ

```
void main()
{
    int m;
    int n = 40;
    int *pn;
    // pn tro den n
}
```

Symbol	Addr	Value
	00000000	
	00000001	
	.	
	.	
n	003DFB0C	40
	.	
	.	
	.	
m	003DFB18	???
	.	
	.	
	.	

```
void main()
```

```
{
```

```
    int m;
```

```
    int n = 40;
```

```
    int* pn; // pn la bien con tro
```

```
    pn = &n; // pn tro den n;
```

```
    printf("&n = %08X pn = %08X\n", &n, pn);
```

```
    printf("n = %d\t*pn = %d\n", n, *pn);
```

```
    m = *pn;
```

```
    printf("m = %d\n", m);
```

```
}
```

Output:

```
&n = 004FFB44 pn = 004FFB38
```

```
n = 40 *pn = 40
```

```
m = 40
```


Con trỏ và địa chỉ

- Ta có thể thao tác trên `*pn` như trên `n`
- Ta cũng có thể gán con trỏ vào con trỏ hoặc khởi động con trỏ bằng con trỏ.

```
(*pn)++;
```

```
*pn += 10;
```

```
++*pn; // ++(*pn)
```

```
m = *pn + 20;
```

```
*pn = *pn * 10;
```

```
m = *pn/2 + 2;
```

```
int *q = pn; // q trỏ đến n
```

Tham số con trỏ

- C chỉ cho phép truyền tham số cho hàm bằng *giá trị* nên ta không thể trực tiếp thay đổi giá trị của biến khi truyền qua hàm.
- Giả sử ta cần viết hàm swap để đổi nội dung 2 số nguyên.
- Ta có thể gián tiếp hoán đổi giá trị của 2 đối tượng bằng cách truyền 2 địa chỉ.

```
void swap(int p1, int p2);
```

```
int main()  
{
```

```
    int a = 10, b = 20;
```

```
    printf("Before: Value of a = %d and value of b = %d\n", a,  
b);
```

```
    swap(a, b);
```

```
    printf("After: Value of a = %d and value of b = %d\n", a, b);
```

```
}
```

```
void swap(int p1, int p2)
```

```
{
```

```
    int t;
```

```
    t = p2;
```

```
    p2 = p1;
```

```
    p1 = t;
```

```
    printf("Value of a (p1) = %d and value of b(p2) = %d\n", p1,  
p2);
```

```
}
```

Truyền bằng giá trị
không làm thay đổi
giá trị tham số.



Output:

Before: Value of a = 10 and value of b = 20

Value of a (p1) = 20 and value of b(p2) = 10

After: Value of a = 10 and value of b = 20

```
void swap(int *p1, int *p2);
```

```
int main()  
{
```

```
    int a = 10, b = 20;
```

```
    printf("Before: Value of a = %d and value of b = %d\n", a,  
b);
```

```
    swap(&a, &b);
```

```
    printf("After: Value of a = %d and value of b = %d\n", a, b);
```

```
}
```

```
void swap(int *p1, int *p2)
```

```
{
```

```
    int t;
```

```
    t = *p2;
```

```
    *p2 = *p1;
```

```
    *p1 = t;
```

```
    printf("Value of a (p1) = %d and value of b(p2) = %d\n", *p1,  
*p2);
```

```
}
```

Giải pháp: Truyền giá trị của con trỏ.



Output:

Before: Value of a = 10 and value of b = 20
Value of a (p1) = 20 and value of b(p2) = 10
After: Value of a = 20 and value of b = 10

```
void swap(int *p1, *int p2);
```

```
int main()
{
    int a = 10, b = 20;

    printf("Before: Value of a = %d and b);
    swap(&a, &b);
    printf("After: Value of a = %d and
}
```

```
void swap(int *p1, *int p2)
{
```

```
    int t;
```

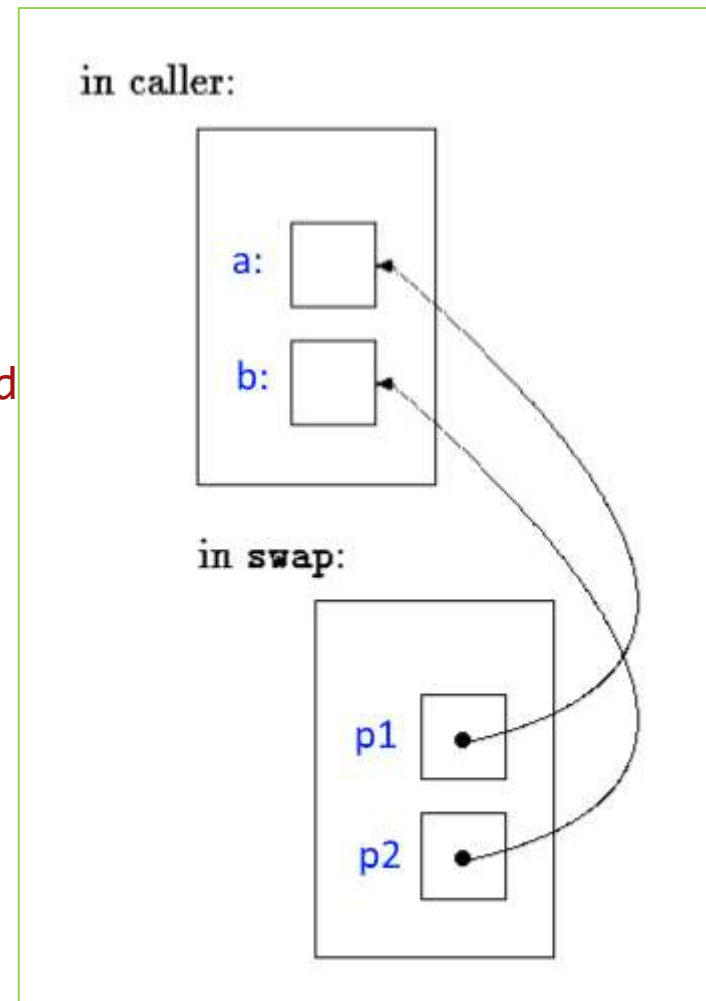
```
    t = *p2;
```

```
    *p2 = *p1;
```

```
    *p1 = t;
```

```
    printf("Value of a (p1) = %d and value of b(p2) = %d\n", *p1,
    *p2);
```

```
}
```



Tham số con trỏ

- Ta truyền tham số con trỏ khi cần thay đổi giá trị của một biến, scanf là hàm thư viện hoạt động theo nguyên tắc này.

Ví dụ:

- Viết hàm cho phép nhập vào các cạnh của một tam giác.

Case study

1. Viết chương trình cho phép nhập vào 2 phân số, xuất ra tổng hiệu, tích, thương của 2 phân số đó.
2. Viết chương trình cho phép nhập vào 3 số, xác định xem ba số đó có tạo thành một tam giác không. Nếu có cho biết đó là tam giác gì, xuất ra chu vi, diện tích tam giác đó.

Tính hợp lệ của con trỏ

- Con trỏ chỉ hợp lệ nếu nó trỏ đến vùng nhớ thuộc quyền kiểm soát của chương trình.
- Dùng con trỏ không hợp lệ có thể dẫn đến những kết quả không kiểm soát được.
- Hai trường hợp con trỏ không hợp lệ thường gặp:
 - Con trỏ không được khởi động, có giá trị rác (giá trị kỳ dị).
 - Con trỏ ban đầu hợp lệ nhưng sau đó không còn hợp lệ.
- Rule of thumb: Khởi động con trỏ bằng giá trị hợp lệ hoặc bằng NULL.

Tính hợp lệ của con trỏ

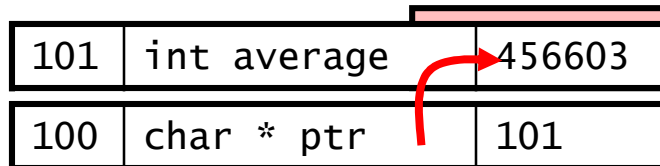
- Con trỏ chỉ hợp lệ nếu nó trỏ đến vùng nhớ thuộc quyền kiểm soát của chương trình.
- Dùng con trỏ không hợp lệ có thể dẫn đến những kết quả không kiểm soát được.
- Hai trường hợp con trỏ không hợp lệ thường gặp:
 - Con trỏ không được khởi động, có giá trị rác (giá trị kỳ dị).
 - Con trỏ ban đầu hợp lệ nhưng sau đó không còn hợp lệ.
- Rule of thumb: Khởi động con trỏ bằng giá trị hợp lệ hoặc bằng NULL.

Không hợp lệ!

Con trỏ đến biến được cấp phát trong stack trở nên không hợp lệ khi biến đó ra khỏi thời gian sống của nó khi stack được pop. Con trỏ sẽ trỏ đến một vùng nhớ sau đó có thể bị dùng lại và bị ghi đè lên.

```
char * get_pointer()
{
→ char x=0;
→ return &x;
}

{
→ char * ptr = get_pointer();
→ *ptr = 12; /* valid? */
→ other_function();
}
```



↑
Grows

Con trỏ và mảng

- Mảng là cấu trúc dữ liệu gồm các phần tử cùng kiểu. Trong bộ nhớ mảng được tổ chức thành danh sách đặc.
- Khai báo mảng:

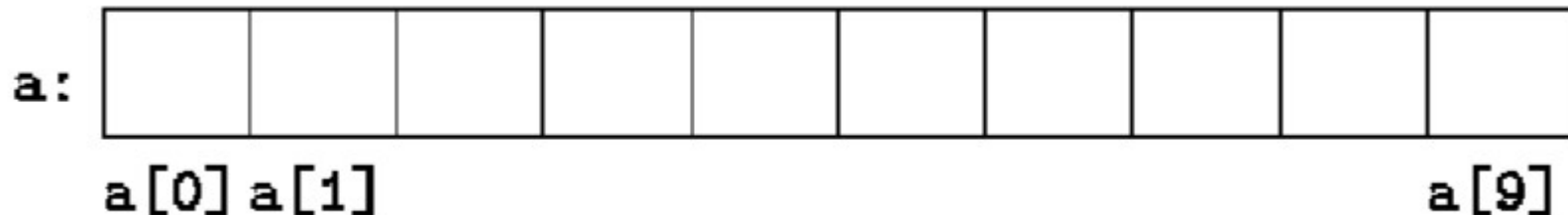
```
int a[10];  
double ar[5];  
char ac[50];
```

Con trỏ và mảng

- Khai báo:

```
int a[10];
```

Sẽ định nghĩa một mảng kích thước 10, nghĩa là một khối mười số nguyên liên tiếp $a[0]$, $a[1]$, ..., $a[9]$. Ký hiệu $a[i]$ tham chiếu đến phần tử thứ i của mảng a .

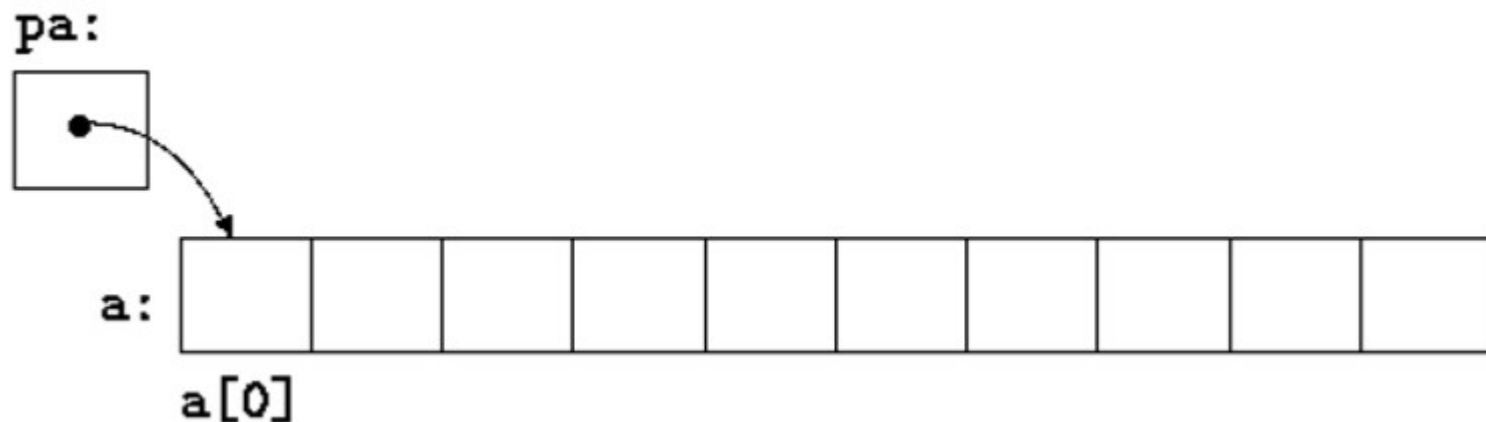


Con trỏ và mảng

- Địa chỉ của phần tử $a[i]$ có thể được truy xuất bằng phép toán lấy địa chỉ &

```
int *pa;
```

```
pa = &a[0]; // pa trỏ đến đầu mảng
```



Con trỏ và mảng

- Ta có thể +, - con trỏ với một số nguyên, nếu `pa` trỏ đến phần tử nào đó trong mảng `a` thì `pa+1` sẽ trỏ đến phần tử kế tiếp, `pa+i` sẽ trỏ đến phần tử cách đó `i` vị trí.

```
int *pa;
```

```
pa = &a[0]; // pa trỏ đến đầu mảng
```

```
pa+1 chứa địa chỉ của a[1],
```

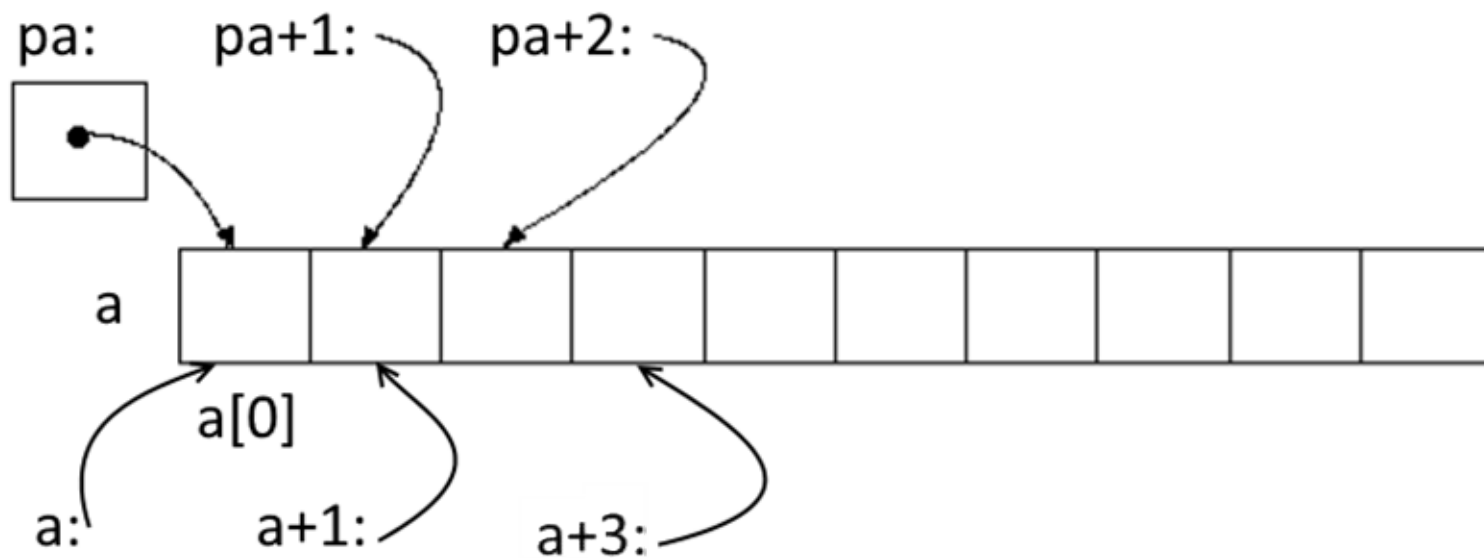
```
pa+i chứa địa chỉ của a[i]
```

Con trỏ và mảng

- `*pa` tham chiếu đến `a[0]`, nghĩa là có nội dung là `a[i]`.
- `*(pa + 1)` tham chiếu đến `a[1]`
- `*(pa + i)` tham chiếu đến `a[i]`
- `a` và `pa` có cùng giá trị. Nghĩa là `a` cũng là con trỏ chỉ đến `a[0]`. (Nhưng `a` là con trỏ hằng)
- Ta có thể viết `*(a+i)` thay cho `a[i]`

Con trỏ và mảng

Tên mảng chứa địa chỉ phần tử đầu tiên của mảng



Tham số mảng

- Khi tên mảng được truyền cho hàm, giá trị được truyền đi thực sự là địa chỉ của phần tử đầu tiên. Vì vậy ta có thể thay đổi giá trị các phần tử của mảng.

```
/* strlen: return length of string s */  
int strlen(char s[])  
{  
    int n;  
    for (n = 0; *s != '\0', s++)  
        n++;  
    return n;  
}
```

Tham số mảng

- Vì tham số mảng là địa chỉ của phần tử đầu tiên. Nên ta có thể *thay đổi* nội dung của mảng. Nói cách khác, mảng được truyền theo phương cách *tham chiếu*.
- Khi khai báo mảng, ta có kích thước *vật lý* của mảng, trong thực tế, ta chỉ dùng kích thước *luận lý* của mảng.
- Sẽ rất *nguy hiểm* nếu ta thao tác ngoài phạm vi *kích thước vật lý* của mảng.
- Tên mảng chỉ cho biết địa chỉ của phần tử đầu tiên, hoàn toàn không có thông tin về kích thước *vật lý* hay *luận lý* của mảng, vì vậy ta phải truyền kích thước *luận lý* của mảng.

Khởi động mảng

- Ta dùng dấu { và } để khởi động mảng.

```
#define dim(a) sizeof(a)/sizeof(a[0])
int main()
{
    int na, nb;
    int a[] = {3, 2, 8, 9, 4};
    double b[] = {1, 9.5, 6.3, 8.4};
    na = dim(a);
    array_output(a, na);
    printf("\n");
    // ...
}
```

```
void array_input(int arr[], int *pn, int max_size)
{
    scanf("%d", pn);
    while (*pn < 0 || *pn > max_size) {
        printf("Nhap so phan tu trong khoang tu %d den %d: ", 0, max_size);
        scanf("%d", pn);
    }
    for (int i = 0; i < *pn; i++)
        scanf("%d", &arr[i]);
}
```

```
void array_output(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        printf("%4d", arr[i]);
}
```

Tham số mảng

```
#define dim(a) sizeof(a)/sizeof(a[0])

int main()
{
    int a[20], na;
    printf("Nhap mang a (so phan tu va cac  
phan tu: ");
    array_input(a, &na, dim(a));
    array_output(a, na);
    printf("\n");
}
```

Tham số mảng: Vận dụng

1. Viết hàm đảo ngược một mảng.
2. Viết hàm đảo ngược một mảng bằng đệ qui (advanced).
3. Viết hàm nối hai mảng (nối mảng b vào mảng a)
4. Viết hàm nối hai mảng để tạo nên mảng mới.

Số học trên con trỏ

- Ta có thể cộng, trừ một con trỏ với một số nguyên. Có thể tăng giảm một con trỏ.
- Có thể trừ hai con trỏ cùng kiểu. Kết quả là số phần tử giữa hai con trỏ đó.
- Có thể cộng dồn, hoặc trừ dồn một con trỏ với một số nguyên.

Số học trên con trỏ

```
int a[10];  
int *pa, *q;  
pa = &a[0];  
q = pa;  
q++;  
q += 2;  
int x = q - pa;
```


Số học trên con trỏ

Ví dụ: Đảo ngược mảng

```
void array_reverse(int arr[], int n)
{
    int *p, *q;
    for (p = arr, q = arr+n-1; p < q; p++, q--)
        swap(p, q);
}
```

Chuỗi - Con trỏ đến ký tự

- Một chuỗi các ký tự là một dãy liên tiếp các ký tự. Trong C, chuỗi là một mảng các ký tự kết thúc bằng ký tự '\0'.
- Chuỗi hằng: **“Hello, world.”**
là một mảng các ký tự:

H	e	l	l	o	,	w	o	r	l	d	.	'\0'
---	---	---	---	---	---	---	---	---	---	---	---	------

Chuỗi - Con trỏ đến ký tự

- Các định nghĩa sau là khác nhau:

```
char amsg[] = "HCM University";  
/* an array */
```

```
char amsg[40] = "HCM University";  
/* an array */
```

```
char *pmsg = "HCM University"; /*  
a pointer */
```

Tham Số Chuỗi

Ví dụ: Sao chép chuỗi

```
void mystrcpy(char s[], char t[])
{
    for (int i = 0; t[i] != '\0'; i++)
        s[i] = t[i];
    s[i] = '\0';
}
```

- *Bài tập:* Viết lại hàm trên dùng con trỏ thay vì dùng chỉ số mảng.

Một số hàm thư viện

`strcpy(s, t)` // sao chép chuỗi t vào s

`strcmp(s, t)` // so sánh 2 chuỗi

`strcat(s, t)` // nối chuỗi t vào s

`strlen(s)` // đếm số kí tự của chuỗi s

`strrev(s)` // đảo ngược chuỗi s

Mảng con trỏ và con trỏ đến con trỏ

- Con trỏ cũng là một kiểu dữ liệu, vì vậy ta có thể khai báo, định nghĩa mảng con trỏ, con trỏ đến con trỏ.

```
void main()
```

```
{
```

Mảng con trỏ



```
    char *aThu[] = { "!", "Chu Nhat", "Thu  
        Hai", "Thu Ba", "Thu Tu", "Thu Nam",  
        "Thu Sau", "Thu Bay" };
```

```
    int t;
```

```
    do
```

```
    {
```

```
        printf("Nhap vao thu (tu 1 den 7):  
        ");
```

```
        scanf("%d", &t);
```

```
    } while (t < 1 || t > 7);
```

```
    printf("%s\n", aThu[t]);
```

```
}
```

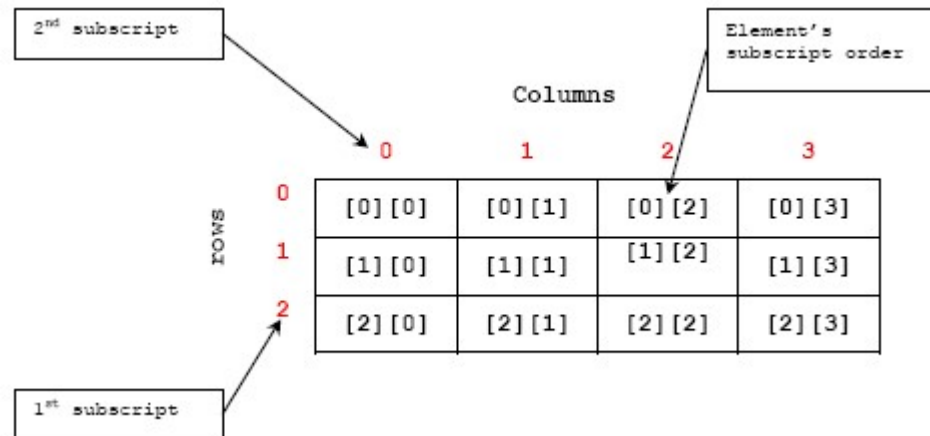
Vận dụng

- Viết chương trình in ra tên các quốc gia trong khu vực Đông Nam Á.
- Viết chương trình in ra tên một số tỉnh thành trong cả nước.
- Viết chương trình in ra tên các tháng trong năm.

Mảng nhiều chiều

- C cho phép tạo mảng nhiều chiều, giúp ta biểu diễn những đại lượng phụ thuộc vào nhiều tham số.
- Mảng 2 chiều thực chất là mảng một chiều mà mỗi phần tử của nó là mảng một chiều. Tương tự cho mảng có số chiều lớn hơn.
- Các phần tử trong mảng nhiều chiều được lưu trữ liên tiếp nhau.

Mảng nhiều chiều



Khởi động mảng nhiều chiều

- Mảng 2 chiều thực chất là mảng 1 chiều mà các phần tử của nó là mảng.
- Ta dùng dấu { và } để khởi động mảng 2 chiều, tương tự cho mảng 3, 4, ... chiều.

Khởi động mảng nhiều chiều

```
int mt[][4] =  
{  
    {1,2,3,4},  
    {4,5,6,7},  
    {9,8,7,6}  
};
```

```
int mt[][4] =  
{  
    1,2,3,4,  
    4,5,6,7,  
    9,8,7,6  
};
```

Tính số ngày, không dùng mảng

```
int SoNgay(int th, int nam)
{
    int songay;

    switch (th) {
        case 2:
            songay = 28 + (nam % 400 == 0 || nam % 4 == 0 && nam % 100
                != 0);
            break;
        case 4: case 6: case 9: case 11:
            songay = 30;
            break;
        case 1: case 3: case 5: case 7: case 8: case 10: case 12:
            songay = 31;
            break;
        default:
            return -1;
    }
    return songay;
}
```

```
bool NamNhuan(int y)
```

Tính số ngày, dùng mảng

```
{  
    return y % 400 == 0 || y % 4 == 0 && y %  
    100 != 0;  
}
```


```
int SoNgay(int m, int y)
```

```
{  
    static int bangNgay[][13] = {  
        { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 },  
        { 0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 }  
    };  
    return bangNgay[NamNhuan(y)][m];  
}
```

So sánh mảng con trỏ và mảng 2 chiều

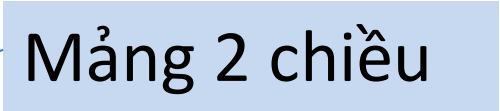
```
void main()  
{  
    char *aThu[] = { "!", "Chu Nhat", "Thu  
        Hai", "Thu Ba", "Thu Tu", "Thu Nam",  
        "Thu Sau", "Thu Bay" };  
    int t;  
    do  
    {  
        printf("Nhap vao thu (tu 1 den 7): ");  
        scanf("%d", &t);  
    } while (t < 1 || t > 7);  
    printf("%s\n", aThu[t]);  
}
```

Mảng con trỏ



So sánh mảng con trỏ và mảng 2 chiều

```
const int MAX = 40;
void main()
{
    char aThu[][MAX] = { "!", "Chu Nhat",
        "Thu Hai", "Thu Ba", "Thu Tu", "Thu
        Nam", "Thu Sau", "Thu Bay" };
    int t;
    do
    {
        printf("Nhap vao thu (tu 1 den 7):
        ");
        scanf("%d", &t);
    } while (t < 1 || t > 7);
    printf("%s\n", aThu[t]);
}
```



Case study

- Dùng mảng nhiều chiều, viết chương trình cho phép nhập vào tên nước, in ra tên thủ đô của nước đó.
- Viết hàm trả về từ cuối của một chuỗi. Áp dụng, viết chương trình cho phép nhập vào một chuỗi đại diện cho họ tên, xuất ra tên tương ứng. (Vd: Vo Van Nam \rightarrow Nam).

Tham số dòng lệnh


```
bool NamNhuan(int y)
{
    return y % 400 == 0 || y % 4 == 0 && y %
    100 != 0;
}

int SoNgay(int m, int y)
{
    static int bangNgay[][13] = {
        { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 },
        { 0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 }
    };
    return bangNgay[NamNhuan(y)][m];
}
```

Tham số dòng lệnh

```
void main(int ac, char *av[])
{
    if (ac < 3)
        exit(printf("Khong du tham so dong
                    lenh.\n"));
    int th = atoi(av[1]),
        nam = atoi(av[2]);

    printf("So ngay trong thang %d nam %d la:
           %d\n", th, nam, SoNgay(th, nam));
}
```



Tham số dòng lệnh

Case study

- Viết chương trình cho phép tạo hoặc nhập một danh sách tên người, sắp xếp lại danh sách theo tên từ nhỏ đến lớn (alphabet). Xuất danh sách sau khi đã sắp xếp.