



DỮ LIỆU MẢNG VÀ KỸ THUẬT XỬ LÝ

NHẬP MÔN LẬP TRÌNH

GVHD: Trương Toàn Thịnh

NỘI DUNG

- Mảng một chiều
- Tham số hàm có dạng là mảng một chiều
- Một số kỹ thuật trên mảng một chiều
- Mảng hai chiều
- Tham số hàm có dạng là mảng hai chiều
- Một số kỹ thuật trên mảng hai chiều
- Chuỗi kí tự

MẢNG MỘT CHIỀU

- Là dãy của các phần tử có cùng kiểu dữ liệu
- Hình ảnh mảng một chiều short

	0	1	2	3	4	5	6	7	8
a	9	8	7	6	5	4	3	2	1
	<100>	<102>	<104>	<106>	<108>	<110>	<112>	<114>	<116>

- Mảng tên a, có 9 phần tử
- Chỉ số tính từ 0 tới 8
- Cú pháp truy cập tới giá trị từng phần tử: $a[\text{<chỉ số>}]$, ví dụ $a[0]$ sẽ là 9.

MẢNG MỘT CHIỀU

- Để tạo một mảng tĩnh ta cần các thông tin:
 - Kiểu của mỗi phần tử
 - Tên biến mảng tĩnh một chiều
 - Số lượng các phần tử mảng (kích thước mảng)
- Cú pháp
`<kiểu_dữ_liệu> tên_mảng[kích_thước]`
- Ví dụ: *bo nho chua 12x4=48byte lien tuc*
 - `int` thang[12]: mảng tên thang chứa 12 phần tử kiểu `int`
 - `double` a[5]: mảng tên a chứa 5 phần tử kiểu `double`

MẢNG MỘT CHIỀU

- Ví dụ

Dòng	Mô tả
1	<code>#include <stdio.h></code>
2	<code>#define N 50</code>
3	<code>void arrayIntInput(int a[N], int& n){</code>
4	<code>while(1){</code>
5	<code>printf("Nhập số phần tử cần dùng < %d", N);</code>
6	<code>scanf("%d", &n);</code>
7	<code>if(n < 0 n > N) printf("Xin nhập lại\n");</code>
8	<code>else break;</code>
9	<code>}</code>
10	<code>for(int i = 0; i < n; i++){</code>
11	<code>printf("Nhập a[%d]: ", i); scanf("%d", &a[i]);</code>
12	<code>}}</code>

MẢNG MỘT CHIỀU

- Chỉ mục không chỉ là hằng số mà còn là biểu thức
- Ví dụ:

```
int a[10], x = 2, y = 3;
```

```
a[(x + y)/2 + 1] = 5; //a[3] = 5
```

- Không được phép gán hai mảng trực tiếp
- Ví dụ:

```
#define N 50
```

```
int a[N], b[N];
```

```
a = b; //Sai
```

- Cần dùng vòng lặp để gán lần lượt từng phần tử:
- Ví dụ

```
for(int i = 0; i < N; i++)
```

```
    a[i] = b[i];
```

MẢNG MỘT CHIỀU

- Với mảng một chiều ta có thể nhập đơn lẻ từng phần tử
- Ví dụ:

```
int a[2];  
scanf("%d", &a[0]);  
scanf("%d", &a[1]);
```

- Nên dùng vòng lặp để nhập dữ liệu mảng
- Ví dụ:

```
int a[2];  
for(int i = 0; i < 2; i++)  
    scanf("%d", &a[i]);
```

MẢNG MỘT CHIỀU

- Với mảng một chiều ta có thể xuất đơn lẻ từng phần tử
- Ví dụ:

```
//...;
```

```
printf(“%d”, a[0]);
```

```
printf(“%d”, a[1]);
```

- Nên dùng vòng lặp để xuất dữ liệu mảng
- Ví dụ:

```
//...;
```

```
for(int i = 0; i < 2; i++)
```

```
    printf(“%d”, a[i]);
```


MẢNG MỘT CHIỀU

- Một số cách khởi gán cho mảng một chiều
- Ví dụ

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

```
int a[5] = {1,2,3};
```

```
int a[5] = {0};
```

```
int a[] = {1,2,3}
```

- Dùng toán tử `sizeof` xác định số phần tử
- Ví dụ

```
int a[] = {1,2,3}, n;
```

```
n = sizeof(a)/sizeof(a[0]);    => n=12/4=3
```

```
for(int i = 0; i < n; i++)
```

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

MẢNG MỘT CHIỀU

- Truyền mảng một chiều cho hàm
 - Khi truyền mảng một chiều có N phần tử cho hàm thì bản chất ta chỉ truyền địa chỉ của phần tử đầu tiên
 - Địa chỉ của phần tử đầu tiên có thể biểu diễn bằng kí hiệu `&a[0]` hay `a`.
 - Giả sử ta có mảng một chiều tĩnh tên là 'a' thì kí hiệu `a` chính là biến **địa chỉ hằng**.
 - Lưu ý: `a` là địa chỉ của `a[0]` nhưng không cần ghi ký hiệu '&'.

MẢNG MỘT CHIỀU

- Truyền mảng một chiều cho hàm

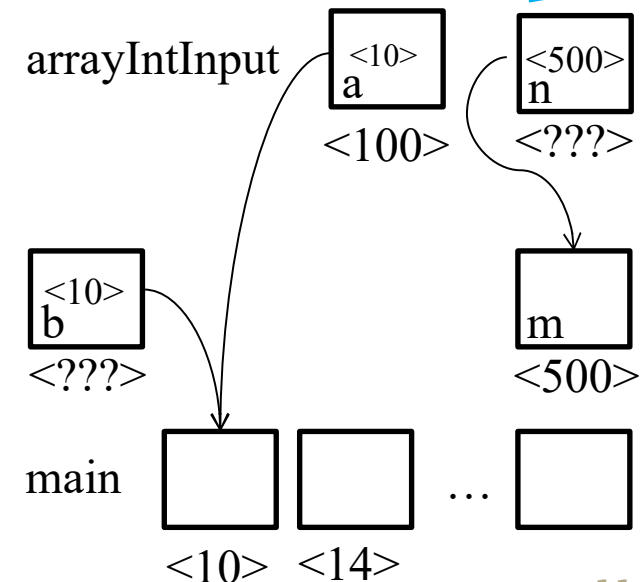
Dòng	Mô tả
1	<code>#include <stdio.h></code>
2	<code>#define N 50</code>
3	<code>void arrayIntInput(int a[N], int& n){...}</code>
4	<code>void arrayIntOutput(int a[N], int n){</code>
5	<code>for(int i = 0; i < n; i++)</code>
6	<code>printf("%d", a[i]);</code>
7	<code>}</code>
8	<code>void main(){</code>
9	<code>int b[N], m;</code>
10	<code>arrayIntInput(b, m);</code>
11	<code>arrayIntOutput(b, m);</code>
12	<code>}</code>

&b -> <10> địa chỉ b đang giữ,
ẩn địa chỉ con trỏ tĩnh b
Lưu ý: &b ≠ b = &b[0]

&a ≠ a = &a[0]

&n ≠ &m

n là con trỏ bị
giấu địa chỉ, n
giữ địa chỉ m



NỘI DUNG

- Mảng một chiều
- Tham số hàm có dạng là mảng một chiều
- Một số kỹ thuật trên mảng một chiều
- Mảng hai chiều
- Tham số hàm có dạng là mảng hai chiều
- Một số kỹ thuật trên mảng hai chiều
- Chuỗi kí tự

MỘT SỐ KỸ THUẬT TRÊN MẢNG MỘT CHIỀU

- Sắp xếp
 - Thuật toán đổi chỗ trực tiếp (“exchange sort”)
 - Đơn giản dễ viết
 - Kém hiệu quả khi kích thước dữ liệu lớn
 - Phù hợp với mức độ nhập môn

Dòng	Mô tả
1	<code>void Sapxep(int a[], int n){</code>
2	<code>for(int i = 0; i < n - 1; i++){</code>
3	<code>for(int j = i + 1; j < n; j++){</code>
4	<code>if(a[i] > a[j]){int temp = a[i]; a[i] = a[j]; a[j] = temp;}</code>
5	<code>}</code>
6	<code>}}</code>

MỘT SỐ KỸ THUẬT TRÊN MẢNG MỘT CHIỀU

- Sắp xếp

$i = 0$	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
	9	6	3	5	8	2	1	4
$j = 1$	6	9						
$j = 2$	3	9	6					
$j = 3$	3	9	6	5				
$j = 4$	3	9	6	5	8			
$j = 5$	2	9	6	5	8	3		
$j = 6$	1	9	6	5	8	3	2	
$j = 7$	1	9	6	5	8	3	2	4

MỘT SỐ KỸ THUẬT TRÊN MẢNG MỘT CHIỀU

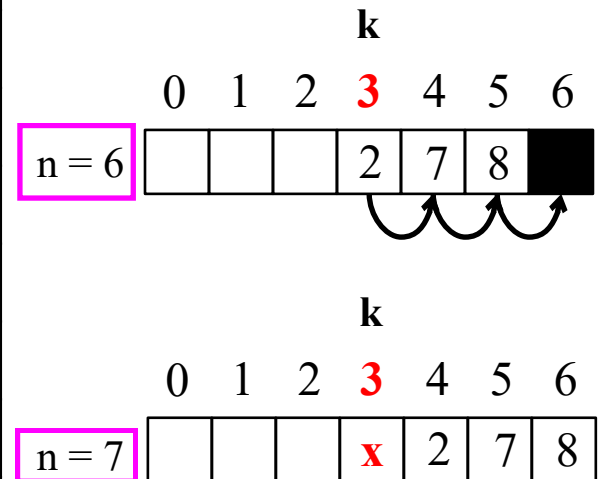
- Sắp xếp

$i = 1$	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
	1	9	6	5	8	3	2	4
$j = 2$	1	6	9					
$j = 3$	1	5	9	6				
$j = 4$	1	5	9	6	8			
$j = 5$	1	3	9	6	8	5		
$j = 6$	1	2	9	6	8	5	3	
$j = 7$	1	2	9	6	8	5	3	4
$i = 6$	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
	1	2	3	4	5	6	9	8
$j = 7$	1	2	3	4	5	6	8	9

MỘT SỐ KỸ THUẬT TRÊN MẢNG MỘT CHIỀU

- Thêm phần tử vào mảng: Thêm số nguyên **x** vào mảng **n** phần tử tại vị trí **k**.

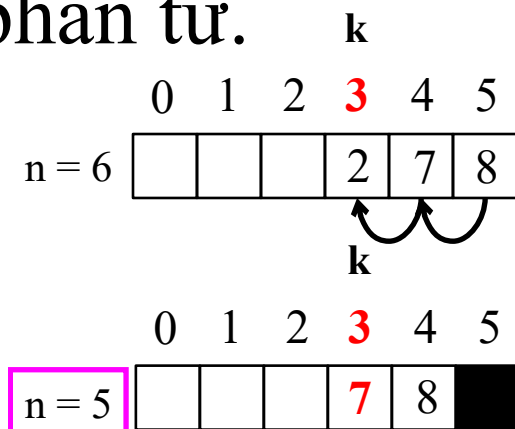
Dòng	Mô tả
1	<code>bool Chen(int a[], int &n, int x, int k){</code>
2	<code>if(k < 0 k > n) return false;</code>
3	<code>for(int i = n - 1; i >= k; i--)</code>
4	<code> a[i+1] = a[i];</code>
5	<code> a[k] = x;</code>
6	<code> n++;</code> ←
7	<code> return true;</code>
8	<code>}</code>



MỘT SỐ KỸ THUẬT TRÊN MẢNG MỘT CHIỀU

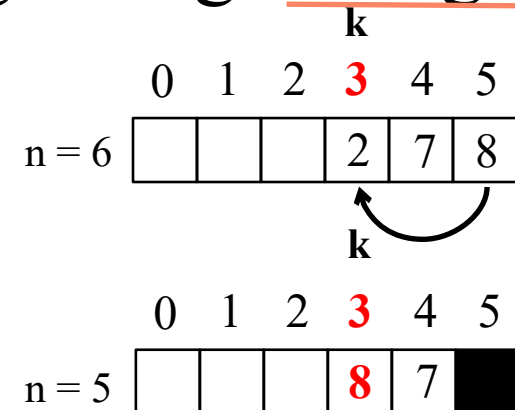
- Xóa một phần tử trong mảng: Xóa phần tử tại vị trí k của mảng n phần tử.

Dòng	Mô tả
1	<code>void Xoa(int a[], int &n, int k){</code>
2	<code>if(k < 0 k > n) return;</code>
3	<code>for(int i = k; i < n - 1; i++) a[i] = a[i + 1];</code>
4	<code>n--;</code> } ←



- Xóa một phần tử trong mảng không duy trì thứ tự

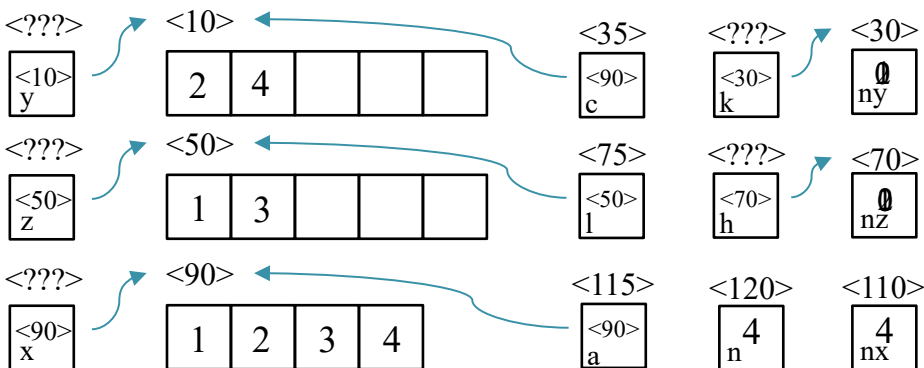
Dòng	Mô tả
1	<code>void Xoa(int a[], int &n, int k){</code>
2	<code>if(k < 0 k > n) return;</code>
3	<code>a[k] = a[n - 1];</code>
4	<code>n--;</code> }



MỘT SỐ KỸ THUẬT TRÊN MẢNG MỘT CHIỀU

- Tách dữ liệu từ một mảng cho hai mảng

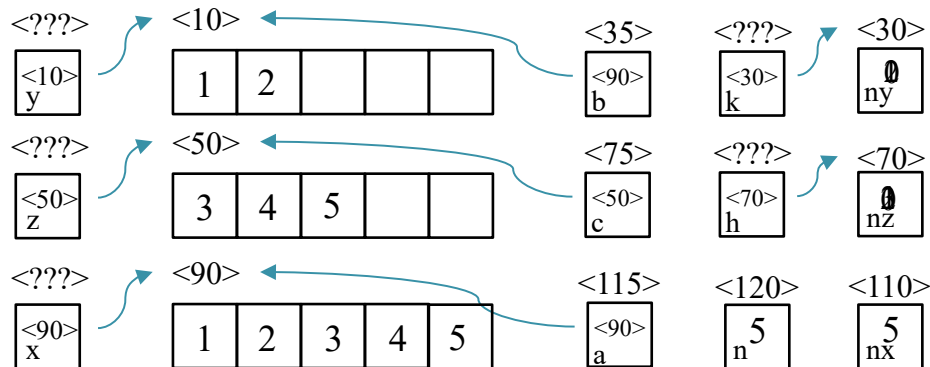
Dòng	Mô tả		
1	<code>void Tachchanle(int a[], int n, int c[], int &k, int l[], int &h){</code>	8	<code>void main(){</code>
2	<code>k = h = 0;</code>	9	<code>int y[5], z[5], ny, nz;</code>
3	<code>for(int i = 0; i < n; i++){</code>	10	<code>int x[] = {1, 2, 3, 4}, nx = 4;</code>
4	<code>if(a[i] % 2 == 0) c[k++] = a[i];</code>	11	<code>Tachchanle(x, nx, y, ny, z, nz);</code>
5	<code>else l[h++] = a[i];</code> ←	12	<code>}</code>
6	<code>}</code>		
7	<code>}</code>		



MỘT SỐ KỸ THUẬT TRÊN MẢNG MỘT CHIỀU

- Tách đôi dữ liệu từ một mảng

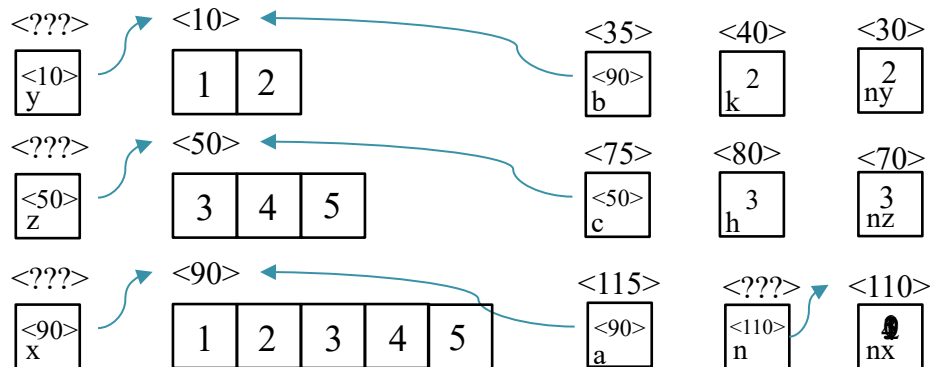
Dòng	Mô tả		
1	<code>void Tachdoi(int a[], int n, int b[], int &k, int c[], int &h){</code>	6	<code>void main(){</code>
2	<code>k = h = 0;</code>	7	<code>int y[5], z[5], ny, nz;</code>
3	<code>for(int i = 0; i < n/2; i++) b[k++] = a[i];</code>	8	<code>int x[] = {1, 2, 3, 4, 5}, nx = 5;</code>
4	<code>for(int j = n/2; j < n; j++) c[h++] = a[j];</code>	9	<code>Tachdoi(x, nx, y, ny, z, nz);</code>
5	<code>}</code>	10	<code>}</code>



MỘT SỐ KỸ THUẬT TRÊN MẢNG MỘT CHIỀU

- Ghép hai mảng vào một mảng

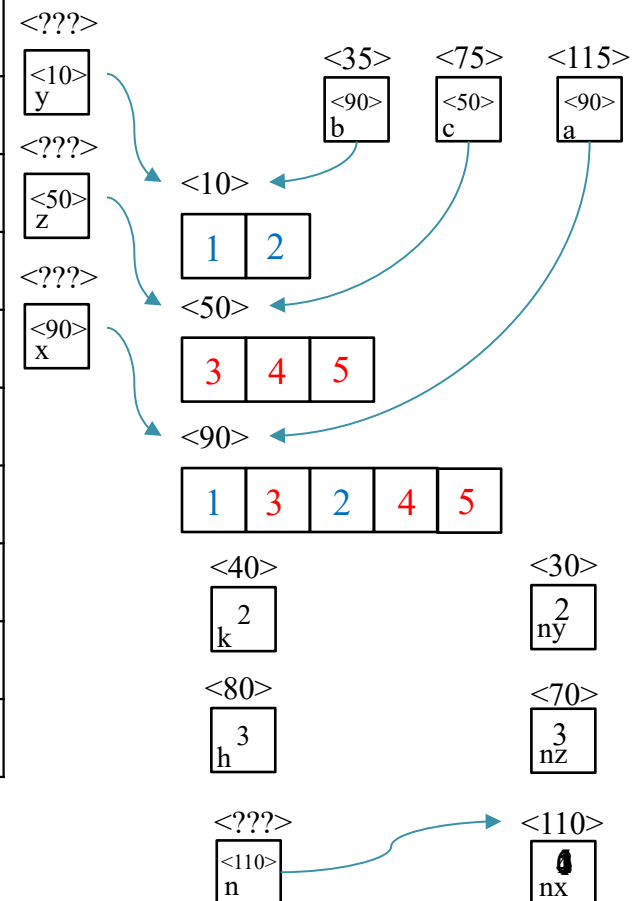
Dòng	Mô tả		
1	<code>void GhepNoi(int a[], int &n, int b[], int k, int c[], int h){</code>	6	<code>void main(){</code>
2	<code>n = 0;</code>	7	<code>int y[] = {1, 2}, z[] = {3, 4, 5};</code>
3	<code>for(int i = 0; i < k; i++) a[n++] = b[i];</code>	8	<code>int ny = 2, nz = 3, nx, x[5];</code>
4	<code>for(int j = 0; j < h; j++) a[n++] = c[j];</code>	9	<code>GhepNoi(x, nx, y, ny, z, nz);</code>
5	<code>}</code>	10	<code>}</code>



MỘT SỐ KỸ THUẬT TRÊN MẢNG MỘT CHIỀU

- Ghép xen kẽ hai mảng vào một mảng

Dòng	Mô tả
1	<code>void GhepXK(int a[], int &n, int b[], int k, int c[], int h){</code>
2	<code> n = 0; int i = 0, j = 0;</code>
3	<code> while((i < k) && (j < h)){</code>
4	<code> if(n % 2 == 0) a[n++] = b[i++];</code>
5	<code> else a[n++] = c[j++];</code>
6	<code> }</code>
7	<code> while(i < k) a[n++] = b[i++];</code>
8	<code> while(j < h) a[n++] = c[j++];</code>
9	<code>}</code>



NỘI DUNG

- Mảng một chiều
- Tham số hàm có dạng là mảng một chiều
- Một số kỹ thuật trên mảng một chiều
- Mảng hai chiều
- Tham số hàm có dạng là mảng hai chiều
- Một số kỹ thuật trên mảng hai chiều
- Chuỗi kí tự

MẢNG HAI CHIỀU

- Là ma trận (**bảng**) của các phần tử có cùng kiểu dữ liệu
- Hình ảnh mảng hai chiều

0	9	8	7
1	6	5	4
2	3	2	1

- Mảng hai chiều tên a, có 9 phần tử
- Chỉ số tính dòng từ 0 tới 2, chỉ số cột từ 0 tới 2
- Cú pháp truy cập tới giá trị từng phần tử: a[<chỉ số dòng>][<chỉ số cột>], ví dụ a[0][1] sẽ là 8.

MẢNG HAI CHIỀU

- Để tạo một mảng tĩnh hai chiều ta cần các thông tin:
 - Kiểu của mỗi phần tử
 - Tên biến mảng tĩnh hai chiều
 - Số dòng và cột ma trận (kích thước ma trận)
- Cú pháp
`<kiểu_dữ_liệu> tên_mảng[kích_thước_dòng][kích_thước_cột]`
- Ví dụ:
`12x11x4byte = [quá trời] byte => [50][50] is ok`
 - `int a[12][11]`: mảng tên a chứa 12×11 phần tử kiểu `int`
 - `double a[5][5]`: mảng tên a chứa 5×5 phần tử kiểu `double`

MẢNG HAI CHIỀU

- Ví dụ

Dòng	Mô tả
1	<code>#include <stdio.h></code>
2	<code>#define MaxRow 20</code>
3	<code>#define MaxCol 20</code>
4	<code>void main(){</code>
5	<code>int a[MaxRow][MaxCol], mRow, nCol, i, j;</code>
6	<code>printf("mRow = "); scanf("%d", &mRow);</code>
7	<code>printf("nCol = "); scanf("%d", &nCol);</code>
8	<code>for(i = 0; i < mRow; i++){</code>
9	<code>for(j = 0; j < nCol; j++)</code>
10	<code>scanf("%d", &a[i][j]);</code>
11	<code>}</code>
12	<code>}</code>

MẢNG HAI CHIỀU

- Chỉ mục không chỉ là hằng số mà còn là biểu thức

- Ví dụ:

```
int a[7][7], x = 2, y = 3;
```

```
a[y - x][y + x] = 5; //a[1][5] = 5
```

- Không được phép gán hai mảng trực tiếp

- Ví dụ:

```
#define R 50
```

```
#define C 50
```

```
int a[R][C], b[R][C];
```

```
a = b; //Sai
```

- Cần dùng vòng lặp để gán lần lượt từng phần tử:

- Ví dụ

```
for(int i = 0; i < R; i++)  
    for(int j = 0; j < C; j++)  
        a[i][j] = b[i][j];
```

MẢNG HAI CHIỀU

- Với mảng hai chiều ta có thể nhập đơn lẻ từng phần tử
- Ví dụ:

```
int a[2][2];  
scanf("%d", &a[0][1]);  
scanf("%d", &a[1][0]);
```

- Nên dùng vòng lặp để nhập dữ liệu mảng hai chiều
- Ví dụ:

```
int a[2][2];  
for(int i = 0; i < 2; i++)  
    for(int j = 0; j < 2; j++)  
        scanf("%d", &a[i][j]);
```

MẢNG HAI CHIỀU

- Với mảng hai chiều ta có thể xuất đơn lẻ từng phần tử
- Ví dụ:

```
//...;
```

```
printf(“%d”, a[0][1]);
```

```
printf(“%d”, a[1][0]);
```

- Nên dùng vòng lặp để xuất dữ liệu mảng hai chiều
- Ví dụ:

```
//...;
```

```
for(int i = 0; i < 2; i++)
```

```
    for(int j = 0; j < 2; j++)
```

```
        printf(“%d”, a[i][j]);
```

MẢNG HAI CHIỀU

- Một số cách khởi gán cho mảng hai chiều
- Ví dụ

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

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

```
int a[3][2] = {1,2,3,4,5}; //Các phần tử  
                           //thiếu sẽ tự động có giá trị zero
```

MẢNG HAI CHIỀU

- Truyền mảng hai chiều cho hàm
 - Chỉ truyền tên mảng tại nơi gọi hàm
 - Trong định nghĩa và khai báo hàm ta chỉ khai báo rõ số lượng cột
 - C xem mảng hai chiều là một mảng các mảng một chiều
 - Ví dụ:

```
#define M 2
```

```
#define N 3      con trỏ cấp 2
```

```
void XuLy(int a[][N], int m, int n){//...}
```

```
void main(){
```

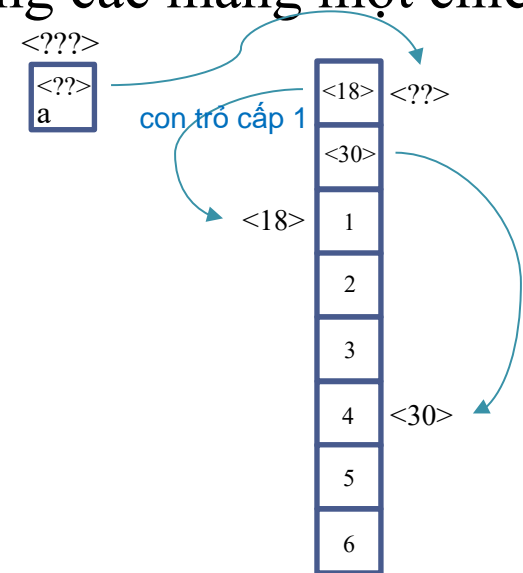
```
    int a[M][N], m, n;
```

```
    //...
```

```
    XuLy(a, m, n);
```

```
    //a[i][j] = (*(a + i) + j) => 4 = *((*a) + 3) = (*(a+1))
```

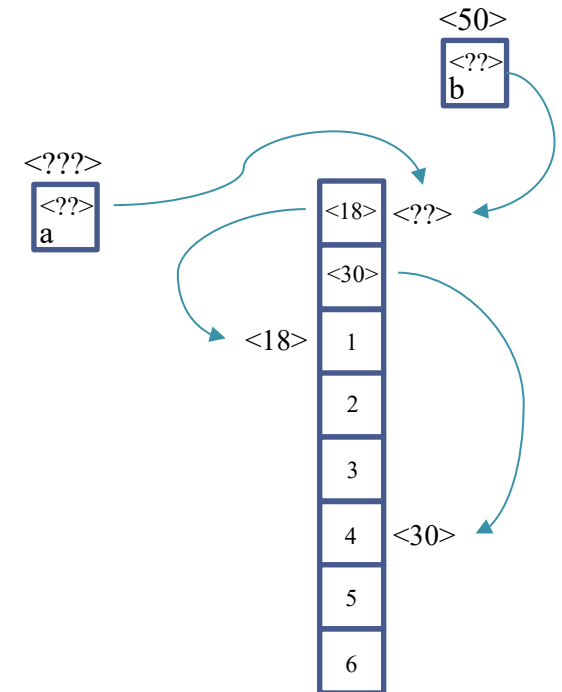
```
}
```



MẢNG HAI CHIỀU

- Truyền mảng hai chiều cho hàm

Dòng	Mô tả
1	<code>#include <stdio.h></code>
2	<code>#define M 20</code>
3	<code>#define N 30</code>
4	<code>void array2DIntInput(int b[][N], int& m, int& n){</code>
5	<code>scanf("%d", &m); // Nhập số dòng</code>
6	<code>scanf("%d", &n); // Nhập số cột</code>
7	<code>for(int i = 0; i < m; i++){</code>
8	<code>for(int j = 0; j < n; j++){</code>
9	<code>scanf("%d", &b[i][j]);</code>
10	<code>}</code>
11	<code>}</code>
12	<code>}</code>



MẢNG HAI CHIỀU

- Truyền mảng hai chiều cho hàm

Dòng	Mô tả
1	//...
2	void array2DOutput(int a[][N], int m, int n){
3	for(int i = 0; i < m; i++){
4	for(int j = 0; j < n; j++){
5	printf("%d", a[i][j]);
6	printf("\n");
7	}
8	}
9	void main(){
10	int a[M][N], m, n;
11	array2DInput(a, m, n);
12	array2DOutput(a, m, n);}

NỘI DUNG

- Mảng một chiều
- Tham số hàm có dạng là mảng một chiều
- Một số kỹ thuật trên mảng một chiều
- Mảng hai chiều
- Tham số hàm có dạng là mảng hai chiều
- Một số kỹ thuật trên mảng hai chiều
- Chuỗi kí tự

MỘT SỐ KỸ THUẬT TRÊN MẢNG HAI CHIỀU

- Sắp xếp tăng

Dòng	Mô tả
1	<code>void sapxep tang(int a[][20], int d, int c) {</code>
2	<code>for (int i=0; i<d*c;i++) {</code>
3	<code>for (int j=0; j<d*c;j++) {</code>
4	<code>if (a[i/c][i%c] < a[j/c][j%c]) {</code>
5	<code>int tmp = a[i/c][i%c] ;</code>
6	<code>a[i/c][i%c] = a[j/c][j%c];</code>
7	<code>a[j/c][j%c] = tmp ;</code>
8	<code>}</code>
9	<code>}</code>
10	<code>}</code>
11	<code>}</code>

Ví dụ:

2 6 1 2
1 5 → 5 6
8 9 8 9

MỘT SỐ KỸ THUẬT TRÊN MẢNG HAI CHIỀU

- Chép nội dung dòng k vào dòng h

Dòng	Mô tả
1	<code>void ChepDong(int a[][20], int m, int n, int k, int h) {</code>
2	<code>if(k<1 k >= m h < 1 h >= m) return;</code>
3	<code>for (int i = 0; i<n; i++) {</code>
4	<code> a[h][i] = a[k][i];</code>
5	<code>}</code>
6	<code>}</code>

Ví dụ xét ma trận (chép nội dung dòng 0 vào dòng 2)

```
2 6 1 2    2 6 1 2
1 5 8 9  → 1 5 8 9
1 4 6 7    2 6 1 2
```

MỘT SỐ KỸ THUẬT TRÊN MẢNG HAI CHIỀU

- Chép nội dung cột k vào cột h

Dòng	Mô tả
1	<code>void ChepCot(int a[][20], int m, int n, int k, int h) {</code>
2	<code>if(k<1 k >= n h < 1 h >= n) return;</code>
3	<code>for (int i = 0; i<m; i++) {</code>
4	<code> a[i][h] = a[i][k];</code>
5	<code>}</code>
6	<code>}</code>

Ví dụ xét ma trận (chép nội dung cột 0 vào cột 3)

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

MỘT SỐ KỸ THUẬT TRÊN MẢNG HAI CHIỀU

- Hoán vị nội dung dòng k và dòng h

Dòng	Mô tả
1	<code>void HoanViDong(int a[][20], int m, int n, int k, int h) {</code>
2	<code>int i, temp;</code>
3	<code>if(k == h) return;</code>
4	<code>for (i = 0; i < n; i++) {</code>
5	<code>temp = a[k][i];</code>
6	<code>a[k][i] = a[h][i];</code>
7	<code>a[h][i] = temp;</code>
8	<code>}</code>
9	<code>}</code>

Ví dụ xét ma trận (hoán vị nội dung dòng 0 và dòng 2)

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

MỘT SỐ KỸ THUẬT TRÊN MẢNG HAI CHIỀU

- Hoán vị nội dung cột k và cột h

Dòng	Mô tả
1	<code>void HoanViCot(int a[][20], int m, int n, int k, int h) {</code>
2	<code>int i, temp;</code>
3	<code>if(k == h) return;</code>
4	<code>for (i = 0; i < m; i++) {</code>
5	<code>temp = a[i][k];</code>
6	<code>a[i][k] = a[i][h];</code>
7	<code>a[i][h] = temp;</code>
8	<code>}</code>
9	<code>}</code>

Ví dụ xét ma trận (hoán vị nội dung cột 0 và cột 3)

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

MỘT SỐ KỸ THUẬT TRÊN MẢNG HAI CHIỀU

- Loại bỏ dòng k (không duy trì thứ tự)

Dòng	Mô tả
1	<code>void LoaiBoDong(int a[][20], int &m, int n, int k) {</code>
2	<code>if((k < 0) (k >= m)) return;</code>
3	<code>if(k != m - 1){</code>
4	<code>ChepDong(a, m, n, m - 1, k);</code>
5	<code>m--;</code>
6	<code>}</code>
7	<code>}</code>

Ví dụ xét ma trận (loại bỏ dòng 0)

2 6 1 2	1 4 6 7	} m
1 5 8 9	→ 1 5 8 9	
1 4 6 7		

MỘT SỐ KỸ THUẬT TRÊN MẢNG HAI CHIỀU

- Loại bỏ dòng k (duy trì thứ tự)

Dòng	Mô tả
1	<code>void LoaiBoDong(int a[][20], int &m, int n, int k) {</code>
2	<code>if((k < 0) (k >= m)) return;</code>
3	<code>if(k == m - 1){ m--; return; }</code>
4	<code>int b[20][20], t = 0;</code>
5	<code>for(int i = 0; i < m; i++){</code>
6	<code>if(i == k) continue;</code>
7	<code>for(int j = 0; j < n; j++) b[t][j] = a[i][j];</code>
8	<code>t++;</code>
9	<code>}</code>
10	<code>for(int i = 0; i < t; i++)</code>
11	<code>for(int j = 0; j < n; j++)</code>
12	<code>a[i][j] = b[i][j];</code>
13	<code>m--; }</code>

Ví dụ xét ma trận (loại bỏ dòng 1)

$$\begin{matrix}
 & \begin{matrix} 2 & 6 & 1 & 2 & 7 \\ 1 & 5 & 8 & 9 & 8 \\ 1 & 4 & 6 & 7 & 3 \\ 5 & 7 & 8 & 2 & 1 \end{matrix} \\
 m \left\{ & \rightarrow & \begin{matrix} 2 & 6 & 1 & 2 & 7 \\ 1 & 4 & 6 & 7 & 3 \\ 5 & 7 & 8 & 2 & 1 \end{matrix}
 \end{matrix}
 \right. m$$

=====Ý tưởng=====

- Tạo ma trận tạm b

- Đổ các dòng (trừ dòng 1) từ a → b

$$\begin{matrix}
 2 & 6 & 1 & 2 & 7 & & 2 & 6 & 1 & 2 & 7 \\
 1 & 5 & 8 & 9 & 8 & \rightarrow & 1 & 4 & 6 & 7 & 3 \\
 1 & 4 & 6 & 7 & 3 & & 5 & 7 & 8 & 2 & 1 \\
 5 & 7 & 8 & 2 & 1 & & & & & &
 \end{matrix}$$

- Đổ ngược lại từ b → a

$$\begin{matrix}
 2 & 6 & 1 & 2 & 7 & & 2 & 6 & 1 & 2 & 7 \\
 1 & 4 & 6 & 7 & 3 & \rightarrow & 1 & 4 & 6 & 7 & 3 \\
 5 & 7 & 8 & 2 & 1 & & 5 & 7 & 8 & 2 & 1 \\
 & & & & & & 5 & 7 & 8 & 2 & 1
 \end{matrix}$$

MỘT SỐ KỸ THUẬT TRÊN MẢNG HAI CHIỀU

- Loại bỏ dòng k (duy trì thứ tự)

Dòng	Mô tả
1	<code>void LoaiBoDong(int a[][20], int &m, int n, int k) {</code>
2	<code>if((k < 0) (k >= m)) return;</code>
3	<code>for(int i = k; i < m - 1; i++) {</code>
4	<code>for(int j = 0; j < n; j++) {</code>
5	<code>a[i][j] = a[i + 1][j];</code>
6	<code>}</code>
7	<code>}</code>
8	<code>m--;</code>
9	<code>}</code>

Ví dụ xét ma trận (loại bỏ dòng 1)

$$\begin{matrix}
 & \begin{matrix} 2 & 6 & 1 & 2 & 7 \\ 1 & 5 & 8 & 9 & 8 \\ 1 & 4 & 6 & 7 & 3 \\ 5 & 7 & 8 & 2 & 1 \end{matrix} \\
 m \left\{ & \rightarrow & \begin{matrix} 2 & 6 & 1 & 2 & 7 \\ 1 & 4 & 6 & 7 & 3 \\ 5 & 7 & 8 & 2 & 1 \end{matrix}
 \end{matrix}
 \right. m$$

=====Ý tưởng=====

- Xác định dòng cần loại bỏ (k = 1)

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

- Chuyển dữ liệu các dòng sau k lên

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

MỘT SỐ KỸ THUẬT TRÊN MẢNG HAI CHIỀU

- Loại bỏ dòng k (duy trì thứ tự)

Dòng	Mô tả
1	<code>void LoaiBoDong(int a[][20], int &m, int n, int k) {</code>
2	<code>if((k < 0) (k >= m)) return;</code>
3	<code>for(int i = k; i < m - 1; i++) {</code>
4	<code>HoanViDong(a, m, n, i, i + 1);</code>
5	<code>}</code>
6	<code>m--;</code>
7	<code>}</code>

Ví dụ xét ma trận (loại bỏ dòng 1)

$$\begin{matrix}
 & \begin{matrix} 2 & 6 & 1 & 2 & 7 \\ 1 & 5 & 8 & 9 & 8 \\ 1 & 4 & 6 & 7 & 3 \\ 5 & 7 & 8 & 2 & 1 \end{matrix} \\
 m \left\{ & \rightarrow & \begin{matrix} 2 & 6 & 1 & 2 & 7 \\ 1 & 4 & 6 & 7 & 3 \\ 5 & 7 & 8 & 2 & 1 \end{matrix}
 \end{matrix}
 \right. m$$

===== Ý tưởng =====

- Xác định dòng cần loại bỏ (k = 1)

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

- Hoán chuyển dữ liệu k với các dòng dưới

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

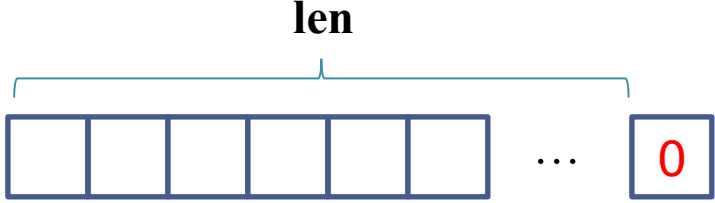
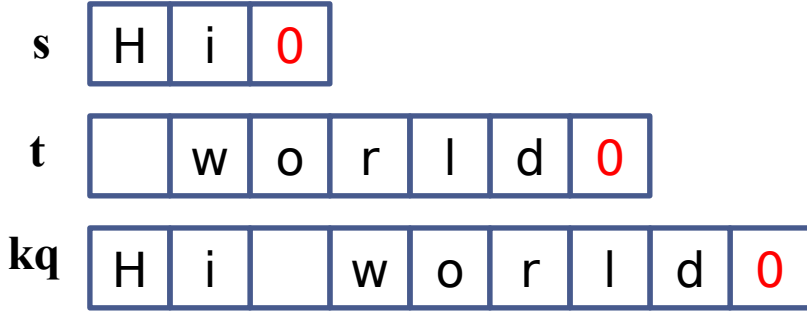
NỘI DUNG

- Mảng một chiều
- Tham số hàm có dạng là mảng một chiều
- Một số kỹ thuật trên mảng một chiều
- Mảng hai chiều
- Tham số hàm có dạng là mảng hai chiều
- Một số kỹ thuật trên mảng hai chiều
- Chuỗi kí tự

CHUỖI KÍ TỰ

- Là mảng các phần tử kiểu `char`
- Kết thúc mảng là kí tự `'\0'`
- Cách khai báo và định nghĩa giá trị:
 - `char s[] = "Nguyễn Van A";`
 - Mảng kí tự tên `s` có 13 phần tử (gồm 12 phần tử nội dung + phần tử `'\0'`)
 - `char s[] = {'h', 'e', 'l', 'l', 'o'};`
- Cách nhập/xuất mảng kí tự
 - `char s[30];`
 - `// tự bỏ 'enter' ra khỏi stdin & tự gắn '\0' vào cuối`
 - `gets_s(s);`
 - `printf("%s\n", s);`

CHUỖI KÍ TỰ

- Một số hàm tiện ích trong <string.h>
 - strlen(char*): trả ra **số lượng kí tự** của chuỗi
 - char s[30];
 - gets_s(s);
 - printf("%d", strlen(s));
 - strcat(char* dest, char* src): nối hai chuỗi dest và src lại với nhau, hàm trả ra **địa chỉ chuỗi dest**.
 - char s[30], t[30], *kq;
 - gets_s(s); gets_s(t);
 - kq = strcat(s, t);
 - printf("%s", kq)

CHUỖI KÍ TỰ

- Một số hàm tiện ích trong <string.h>

- strchr(const char* src, int ch): trả ra con trỏ đến vị trí đầu tiên của chuỗi src chứa ký tự ch

- // con trỏ trỏ tới hằng: không thay đổi được giá trị

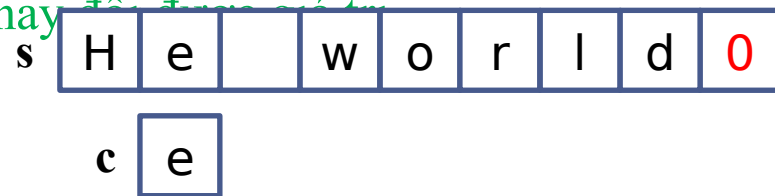
- const char *s = "He world";

- // hằng con trỏ (char* const s): thay đổi hằng giá trị

- int ch = 101;

- char* c = strchr(s, ch);

- if(c != NULL) printf("%c", *c);



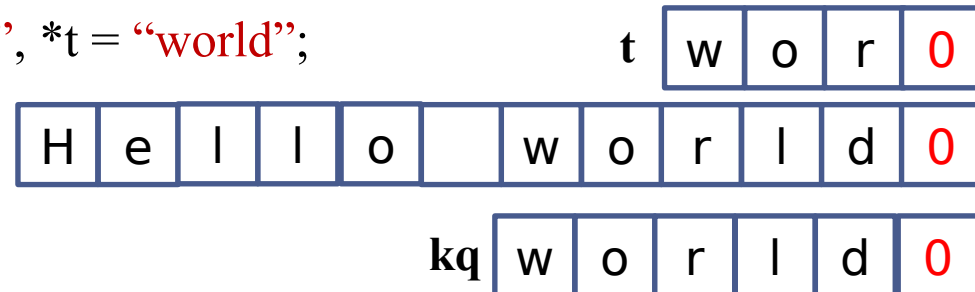
- strstr(const char* mainStr, const char* subStr): trả ra con trỏ đến vị trí đầu tiên của chuỗi subStr trong chuỗi mainStr.

- char *s = "hello world", *t = "world";

- char *kq = strstr(s, t);

- if (kq != NULL)

- printf("%s", kq);



CHUỖI KÍ TỰ

- Một số hàm tiện ích trong <string.h>
 - strtok(char* s, char* delim): tách các ‘từ’ trong chuỗi s (việc tách dựa vào chuỗi delim). Mỗi lần gọi hàm này trả ra con trỏ trỏ tới ‘từ’ tách ra
 - char *s = “They are dogs, cats. The dogs”;
 - char *sep = “ ,.”;
 - char* w = strtok(s, sep);
 - while(w!=NULL){
 - printf(“%s\n”, w);
 - w = strtok(NULL, sep);
 - }

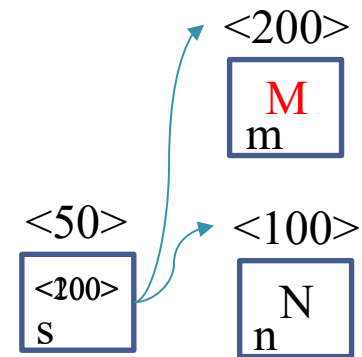
CHUỖI KÍ TỰ

- Phân biệt con trỏ hằng và hằng con trỏ

- Con trỏ hằng: `const char*` s;

- Ví dụ:

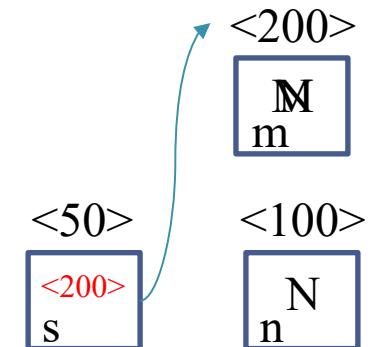
- `char` m = 'M', n = 'N';
 - `const char*` s = &m;
 - `*s = n; // Lỗi`
 - `s = &n; // OK`



- Hằng con trỏ: `char*` `const` s;

- Ví dụ:

- `char` m = 'M', n = 'N';
 - `char*` `const` s = &m;
 - `*s = n; // OK`
 - `s = &n; // Lỗi`



BÀI TẬP

- Bài tập 1:
 - Xây dựng hàm nhập và xuất mảng một chiều các số nguyên
 - Xây dựng hàm tính tổng các phần tử chẵn trong mảng một chiều
 - Xây dựng hàm tính tích các phần tử tại vị trí lẻ trong mảng một chiều

BÀI TẬP

- Bài tập 2:
 - Xây dựng hàm nhập và xuất mảng hai chiều
 - Xây dựng hàm xoay biên trái của ma trận

