

Chương 5

DỮ LIỆU KIỂU MẢNG

Nội dung

1. **Mảng một chiều**
2. **Mảng hai chiều**

5.1. Kiểu dữ liệu mảng một chiều

1. Khái niệm mảng một chiều
2. Sử dụng mảng trong C
3. Xử lý mảng một chiều trong C

5.1.1. Khái niệm mảng một chiều

- **Khái niệm:**

- Mảng một chiều là một dãy liên tiếp các biến có cùng kiểu dữ liệu.

- **Khai báo:**

<Kiểu_dữ_liệu> <Tên_mảng>[<Kích_thước>];

- Kiểu dữ liệu: có thể là các kiểu cơ sở, cấu trúc
- Tên mảng: Một tên phù hợp với quy tắc
- Kích thước: Số nguyên dương

Khái niệm mảng một chiều (tt)

- Ví dụ:

```
#define SIZE 10  
int a[SIZE];
```

Mảng a

0	1	2	3	4	5	6	7	8	9

- Các phần tử mảng

- Các phần tử được xác định bởi tên mảng và chỉ số.
- Chỉ số của phần tử bắt đầu từ 0 đến $\text{SIZE} - 1$ với SIZE là kích thước mảng.
- Ký hiệu phần tử: $\text{tên_mảng}[\text{chỉ_số}]$
- Ví dụ: $a[0]$, $a[1]$, $a[2]$, $a[3]$, ..., $a[9]$ lần lượt là các phần tử của mảng a.

5.1.2. Sử dụng mảng một chiều

- Mảng một chiều được sử dụng để lưu trữ danh sách.
- Hai mảng có cùng kiểu và cùng kích thước cũng không được xem là tương đương nhau, vì thế không thể gán trực tiếp một mảng cho một mảng khác.
- Không thể gán trị cho toàn bộ mảng, chỉ được gán trị cho từng phần tử của mảng.
- Các thao tác xử lý mảng cần được thiết kế thuật toán chi tiết.

5.1.2.1. Đưa dữ liệu vào mảng

- Cách 1: Khởi tạo trong lệnh khai báo

```
int a[SIZE] = {3, -5, 1, 9, 2, 8, 6};
```

- Cách 2: Gán lần lượt từng phần tử

```
void khoiTao(int a[SIZE], int n) {  
    a[0] = 3; a[1] = -5;  
    a[2] = 1; a[3] = 9;  
    a[4] = 2; a[5] = 8;  
    a[6] = 6;  
}
```

Mảng a	0	1	2	3	4	5	6	7	8	9
	3	-5	1	9	2	8	6			

Đưa dữ liệu vào mảng (tt)

- Cách 3: Gán bằng vòng lặp

```
void khoiTao(int a[SIZE], int n) {  
    int i;  
    for (i = 0; i < n; i++)  
    {  
        a[i] = i * i;  
    }  
}
```

Mảng a
n = 7



0	1	2	3	4	5	6	7	8	9
0	1	4	9	16	25	36			

Đưa dữ liệu vào mảng (tt)

- Cách 4: Nhập mảng từ bàn phím.

```
void nhapMang(int a[SIZE], int n)
{
    int i;
    for (i = 0; i < n; i++)
    {
        printf("\ta[%d] = ", i);
        scanf("%d", &a[i]);
    }
}
```

Mảng a
n = 7

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

5.1.2.2. Hiển thị mảng lên màn hình

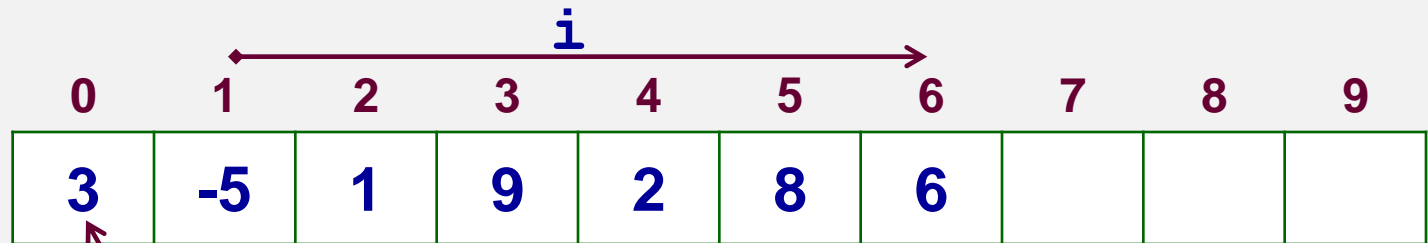
```
void hienThiMang(int a[SIZE], int n)
{
    int i;
    for (i = 0; i < n; i++)
    {
        printf("%d\t", a[i]);
    }
    printf("\n");
}
```

5.1.3. Xử lý mảng một chiều

1. Tìm kiếm
2. Tính toán và thống kê
3. Xóa dữ liệu trong mảng
4. Chèn dữ liệu vào mảng
5. Sắp xếp mảng
6. Ghép mảng, tách mảng

5.1.3.1. Kỹ thuật tìm kiếm

Mảng a
n = 7



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

```
/* Tìm giá trị lớn nhất trong mảng*/  
int max(int a[SIZE], int n){  
    int i, m = a[0];  
    for (i = 1; i < n; i++) {  
        if (m < a[i])  
            m = a[i];  
    }  
    return m;  
}
```

5.1.3.2. Tính toán và thống kê số liệu

Mảng a
n = 7

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

/*Tính giá trị trung bình cộng của các phần tử mảng*/

```
float tbc(int a[SIZE],int n) {  
    int i, tong = 0;  
    for (i = 0; i < n; i++) {  
        tong = tong + a[i];  
    }  
    return (float) tong/n;  
}
```

Tính toán và thống kê số liệu (tt)

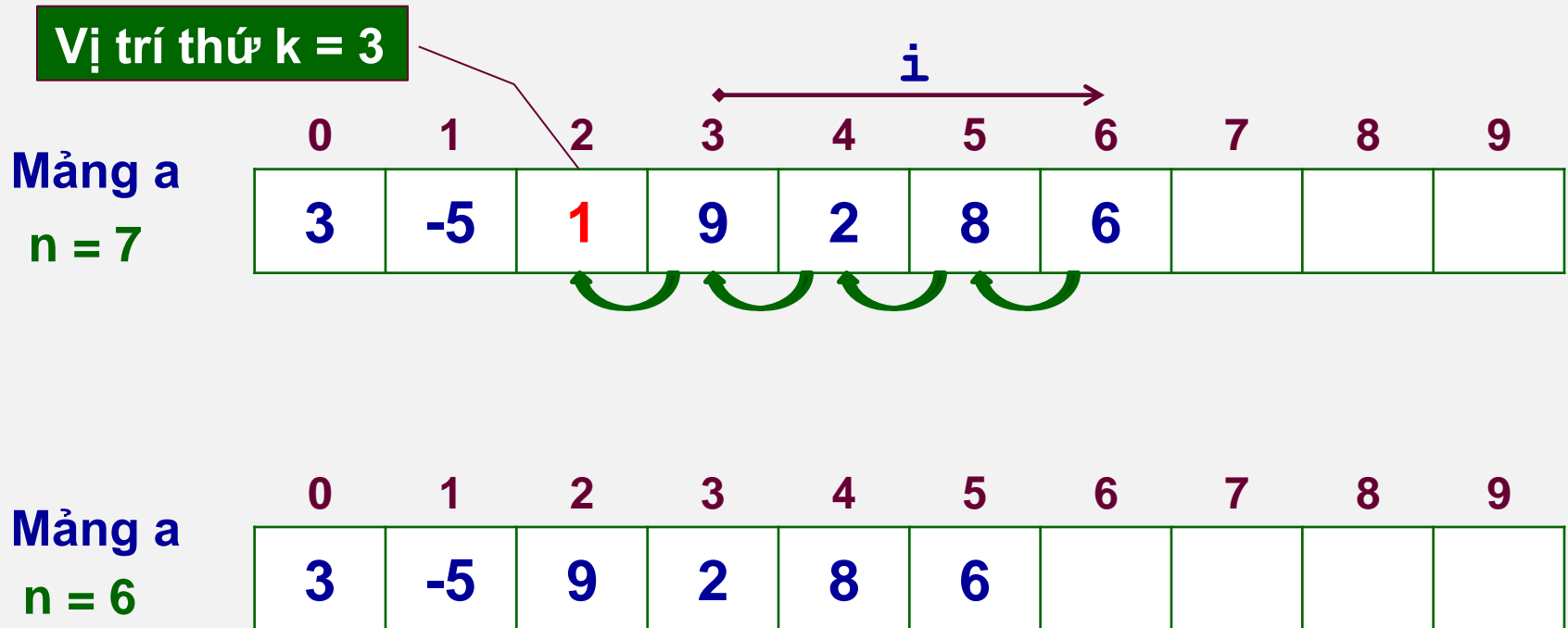
Mảng a
n = 7

<div><div>i</div><div>→</div></div>									
0	1	2	3	4	5	6	7	8	9
3	-5	1	9	2	8	6			

```
/*Tính giá trị tb cộng của các phần tử dương*/  
void tbcSoDuong(int a[SIZE],int n) {  
    int t = 0, d = 0, i;  
    for (i = 0; i < n; i++)  
        if (a[i] > 0) {  
            t = t + a[i];  
            d++;  
        }  
    if (d > 0)  
        printf("\nTBC so duong: %f", (float)t/d);  
    else printf("\nMang khong co so duong nao");  
}
```

5.1.3.3. Xóa một phần tử dữ liệu trong mảng

- Xóa phần tử thứ $k = 3$ trong mảng a đang có n phần tử dữ liệu.



5.1.3.3. Xóa một phần tử dữ liệu trong mảng

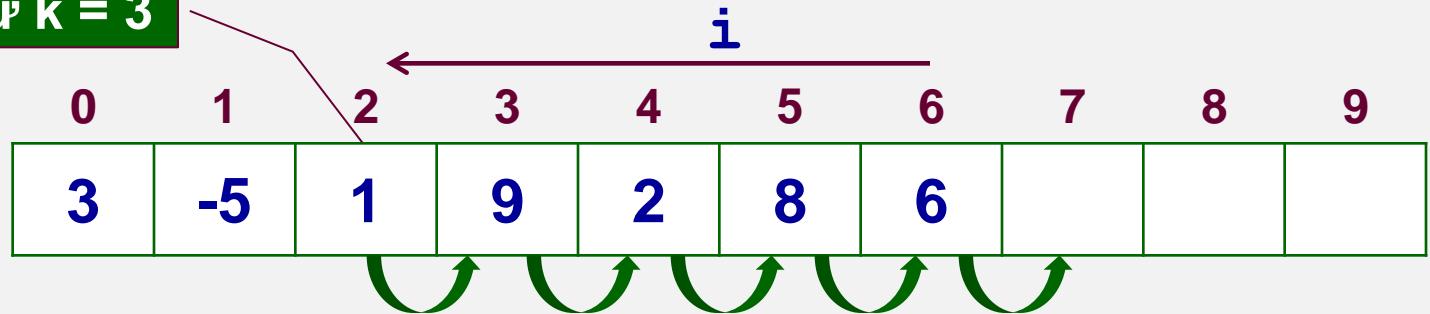
```
void removeElem(int a[SIZE], int *n, int k)
{
    int i;
    for (i = k; i < *n; i++) {
        a[i-1] = a[i];
    }
    *n = *n - 1;
}
```


5.1.3.5. Chèn một phần tử dữ liệu vào mảng

- Chèn phần tử $x = 12$ vào vị trí thứ $k = 3$ trong mảng a .

Vị trí thứ $k = 3$

Mảng a
 $n = 7$



x
12
Mảng a
 $n = 7$



Mảng a
 $n = 8$

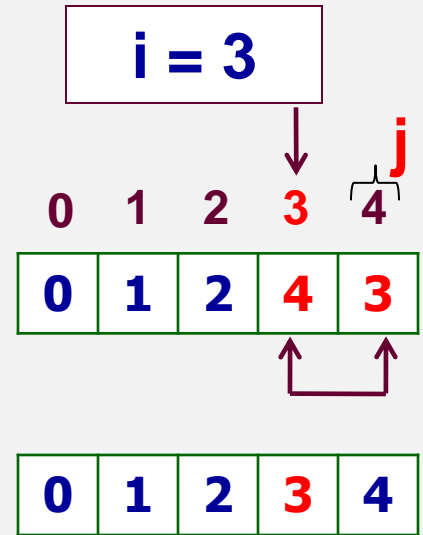
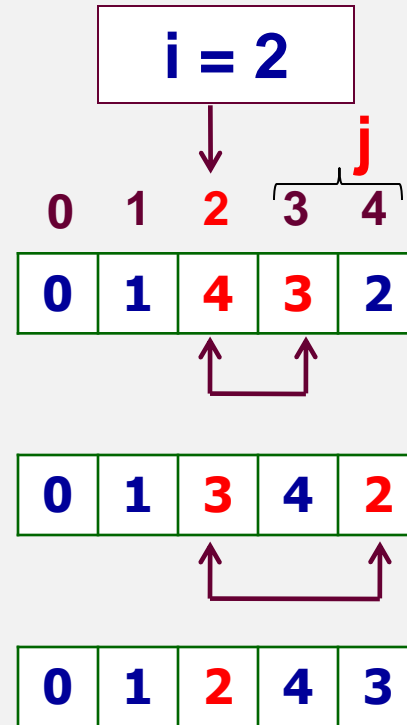
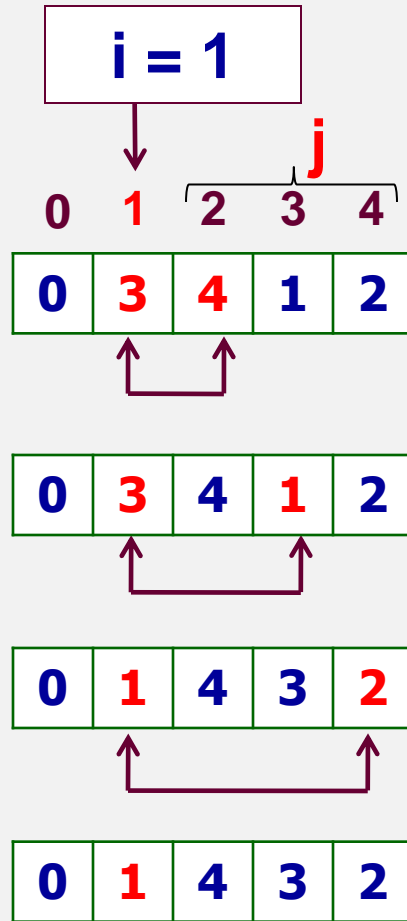
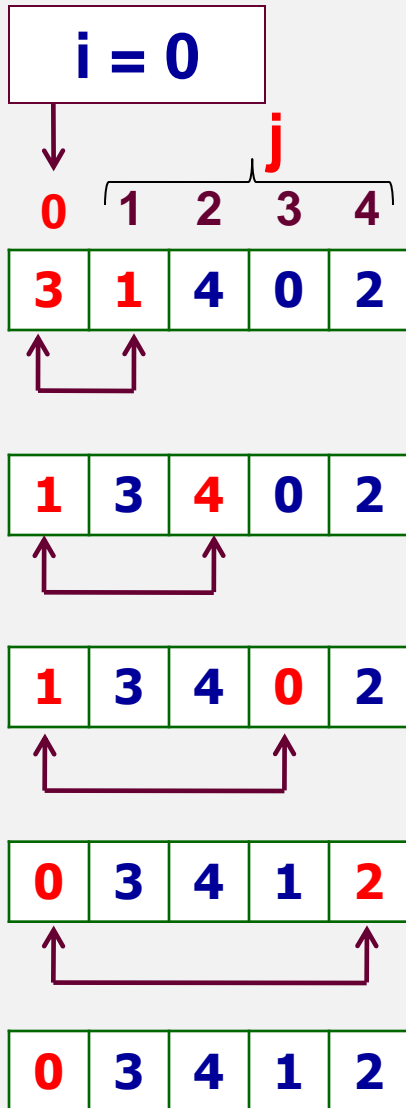


Chèn một phần tử dữ liệu vào mảng (tt)

/*Chèn số x vào vị trí thứ k trong mảng a đang có n phần tử*/

```
void insert(int a[SIZE], int *n, int k, int x)
{
    int i;
    for (i = *n-1; i >= k-1; i--) {
        a[i+1] = a[i];
    }
    a[k-1] = x;
    *n = *n+1;
}
```

5.1.3.5. Sắp xếp dữ liệu trong mảng



Sắp xếp dữ liệu trong mảng (tt)

```
/*Sắp xếp mảng theo chiều tăng dần*/  
void sapXep(int a[SIZE], int n) {  
    int i, j, tg;  
    for (i = 0; i < n-1; i++)  
        for (j = i+1; j < n; j++)  
            if (a[i] > a[j]) {  
                tg = a[i];  
                a[i] = a[j];  
                a[j] = tg;  
            }  
}
```

5.1.5. Bài tập

1. Cài đặt chương trình thực hiện các yêu cầu:
 - Nhập số nguyên dương n thỏa mãn $1 \leq n \leq 30$.
 - Nhập vào một mảng n số nguyên.
 - Hiển thị mảng lên màn hình.
 - Tìm và in ra màn hình giá trị nhỏ nhất, và vị trí đầu tiên mà nó xuất hiện trong mảng.
 - Nhập số nguyên k thỏa mãn $1 \leq k \leq n$.
 - Xóa phần tử thứ k trong mảng, in mảng vừa xóa lên màn hình.

Bài tập

2. Cài đặt chương trình thực hiện các yêu cầu:

- Nhập số nguyên dương n thỏa mãn $1 \leq n \leq 30$.
- Nhập vào một mảng n số thực.
- Hiển thị mảng lên màn hình.
- Tính và in ra màn hình giá trị trung bình cộng của các số âm trong mảng.
- Sắp xếp mảng theo chiều giảm dần, in mảng vừa sắp ra màn hình.

Bài tập

3. Cài đặt chương trình thực hiện các yêu cầu:

- Nhập số nguyên dương n thỏa mãn $1 \leq n \leq 30$.
- Nhập vào một mảng n số nguyên.
- Hiển thị mảng lên màn hình.
- Cho biết mảng có được sắp xếp theo chiều tăng dần hay không?
- Nhập số nguyên k thỏa mãn $1 \leq k \leq n$, và số nguyên x bất kỳ.
- Chèn số nguyên x vào vị trí thứ k trong mảng, in mảng vừa chèn lên màn hình.