Chương 4

Kỹ thuật xử lý mảng

Giảng viên: Nguyễn Chiến Thắng
Email: thangnc.haui@gmail.com

Nội dung

1. Kỹ thuật xử lý mảng một chiều
2. Kỹ thuật xử lý mảng 2 chiều

4.1. Kỹ thuật xử lý mảng một chiều

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

4.1.1. Khái niệm mảng một chiều

- Khái niệm:
  - Mảng một chiều là một dãy liên tiếp các biến có cùng kiểu dữ liệu.
- Khai báo:

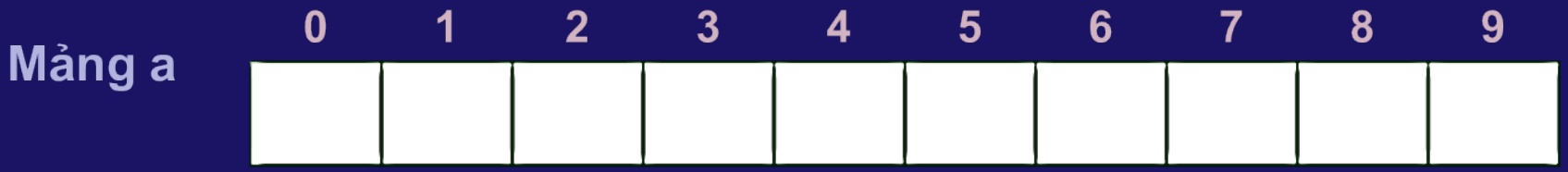
```
<Kiểu_dữ_liệu> <Tên_mảng>[<Kích_thước>];
```

  - <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];
```



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

4.1.2. Sử dụng mảng một chiều

1. Mảng một chiều được sử dụng để lưu trữ danh sách.
2. 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.
3. Không thể thực hiện lệnh gán trị cho toàn bộ mảng, chỉ thực hiện được gán trị cho từng phần tử của mảng.
4. Các thao tác xử lý mảng cần được thiết kế thuật toán chi tiết.

4.1.2.1. Khởi tạo dữ liệu cho mảng

- Giả sử chương trình cần xử lý danh sách số nguyên {3, -5, 1, 9, 2, 8, 6}. Khi đó danh sách được đưa vào mảng theo cách sau:
- Cách 1: Khởi tạo dữ liệu trong lệnh khai báo

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

- Cách 2: Khởi tạo bằng cách gán lần lượt từng phần tử

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

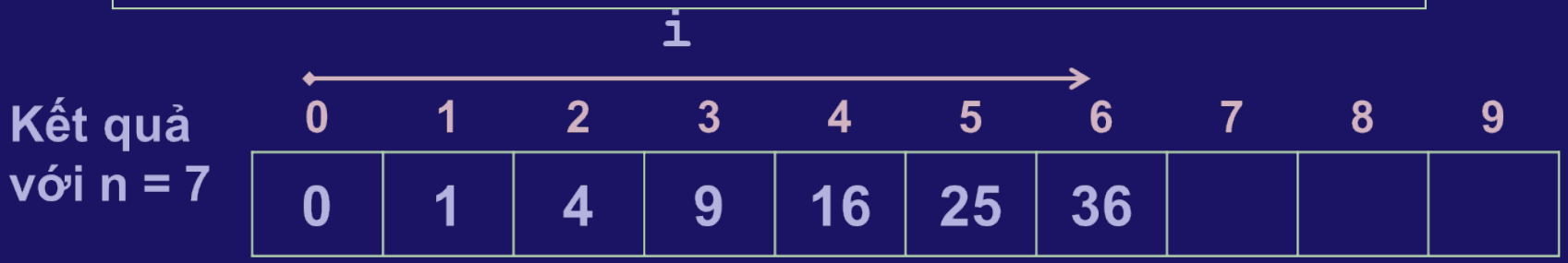
Kết quả

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

Khởi tạo dữ liệu cho mảng (tt)

- Cách 3: Sử dụng vòng lặp nếu dữ liệu theo luật.

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



Khởi tạo dữ liệu cho mảng (tt)

- Cách 4: Dữ liệu là số được sinh ngẫu nhiên

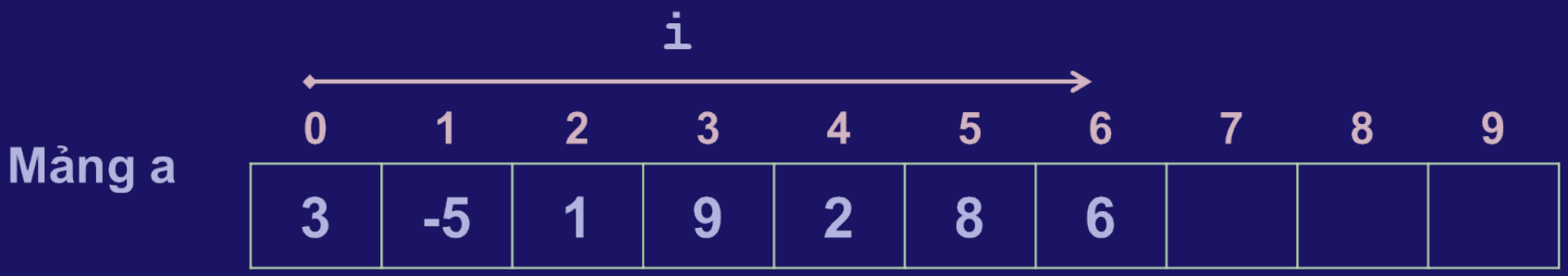
```
void khoiTao(int a[SIZE], int n) {
   srand(int(time(0)); (1)
   for (int i = 0; i < n; i++) {
      //a[i] = rand();
      a[i] = -10 + rand() % (10 + 1 - (-10));
   }
}
```

- Hàm rand() sinh số ngẫu nhiên kiểu int.
- (1) khởi tạo bộ số ngẫu nhiên ở thời điểm hiện tại.
- Các phần tử dữ liệu mảng nhận được là các số ngẫu nhiên trong đoạn [-10, 10].

Khởi tạo dữ liệu cho mảng (tt)

- Cách 5: Nhập mảng từ bàn phím.

```
void nhapMang(int a[SIZE], int n)
{
      for (int i = 0; i < n; i++)
      {
            cout<<"a["<<i<<"] = ";
            cin>>a[i] ;
      }
}
```



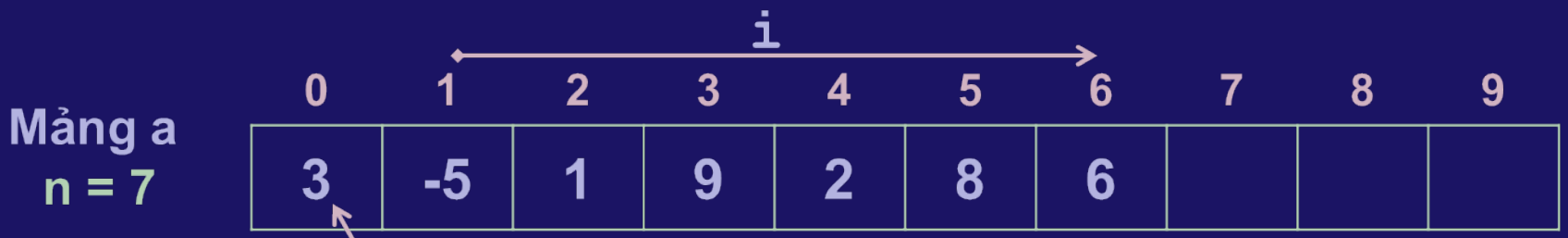
4.1.2.2. Hiển thị mảng lên màn hình

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

4.1.3. Kỹ thuật xử lý mảng một chiều

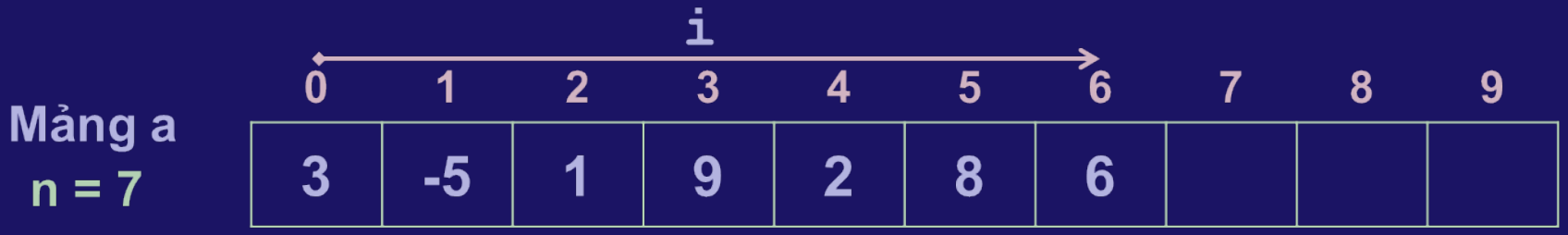
1. Tìm kiếm
2. Tính toán và thống kê
3. Xóa dữ liệu trong mảng
4. Chèn dữ liệu vào mảng
5. Sắp xếp mảng
6. Ghép mảng, tách mảng

4.1.3.1. Kỹ thuật tìm kiếm



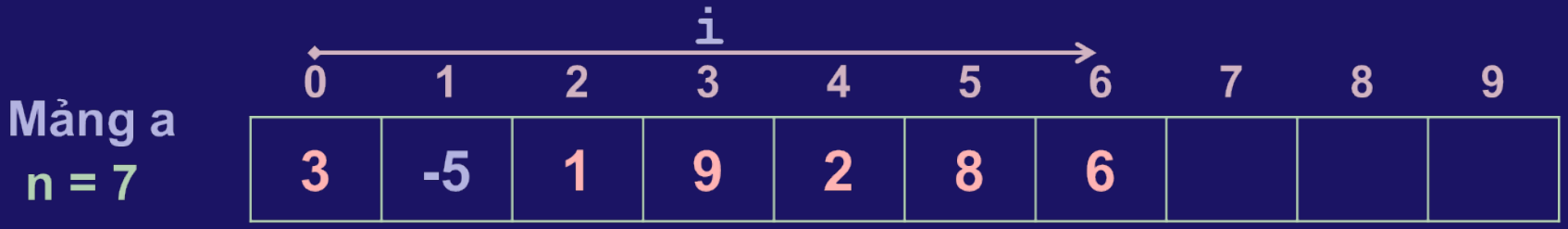

```
/* Tìm giá trị lớn nhất  trong mảng*/
int max(int a[SIZE], int n){
   int m = a[0];
   for (int i = 1; i < n; i++) {
        if (m < a[i])
             m = a[i];
   }
   return m;
}
```

4.1.3.2. Tính toán và thống kê số liệu




```
/*Tính giá trị trung bình cộng của các phần
tử mảng*/
float tbc(int a[SIZE],int n) {
    int tong = 0;
    for (int i = 0; i < n; i++) {
        tong = tong + a[i];
    }
    return (float)tong/n;
}
```

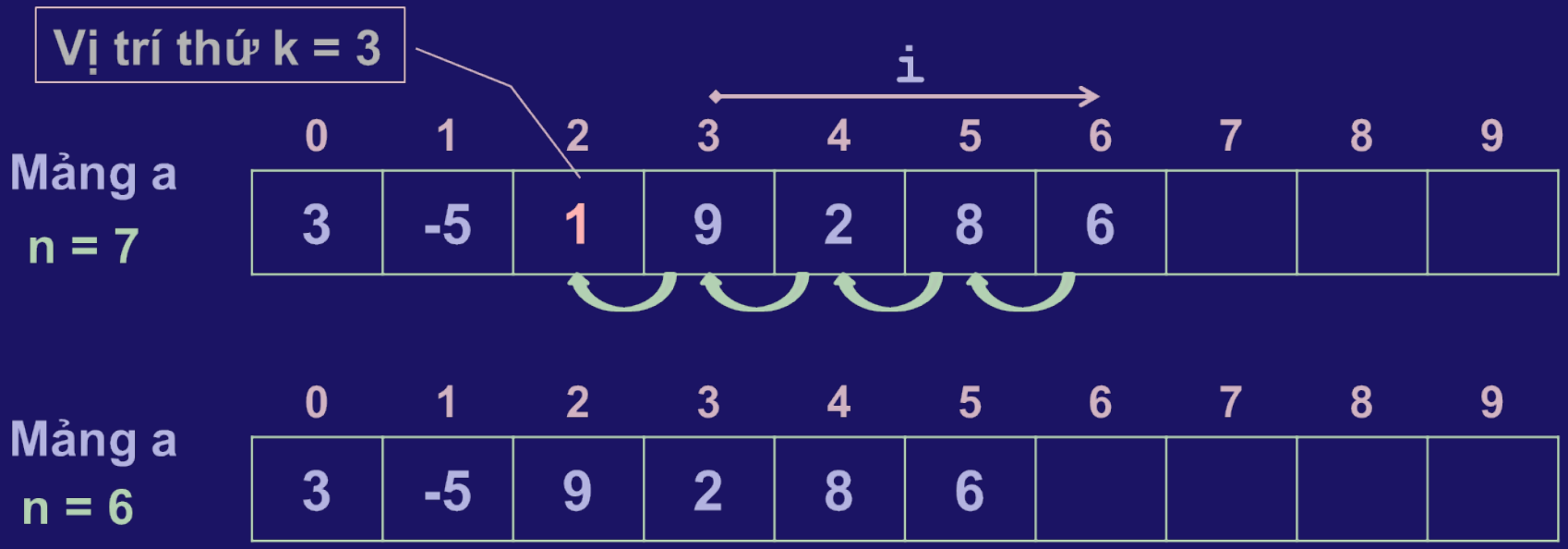
Tính toán và thống kê số liệu (tt)




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

4.1.3.3. Xóa một phần tử dữ liệu trong mảng

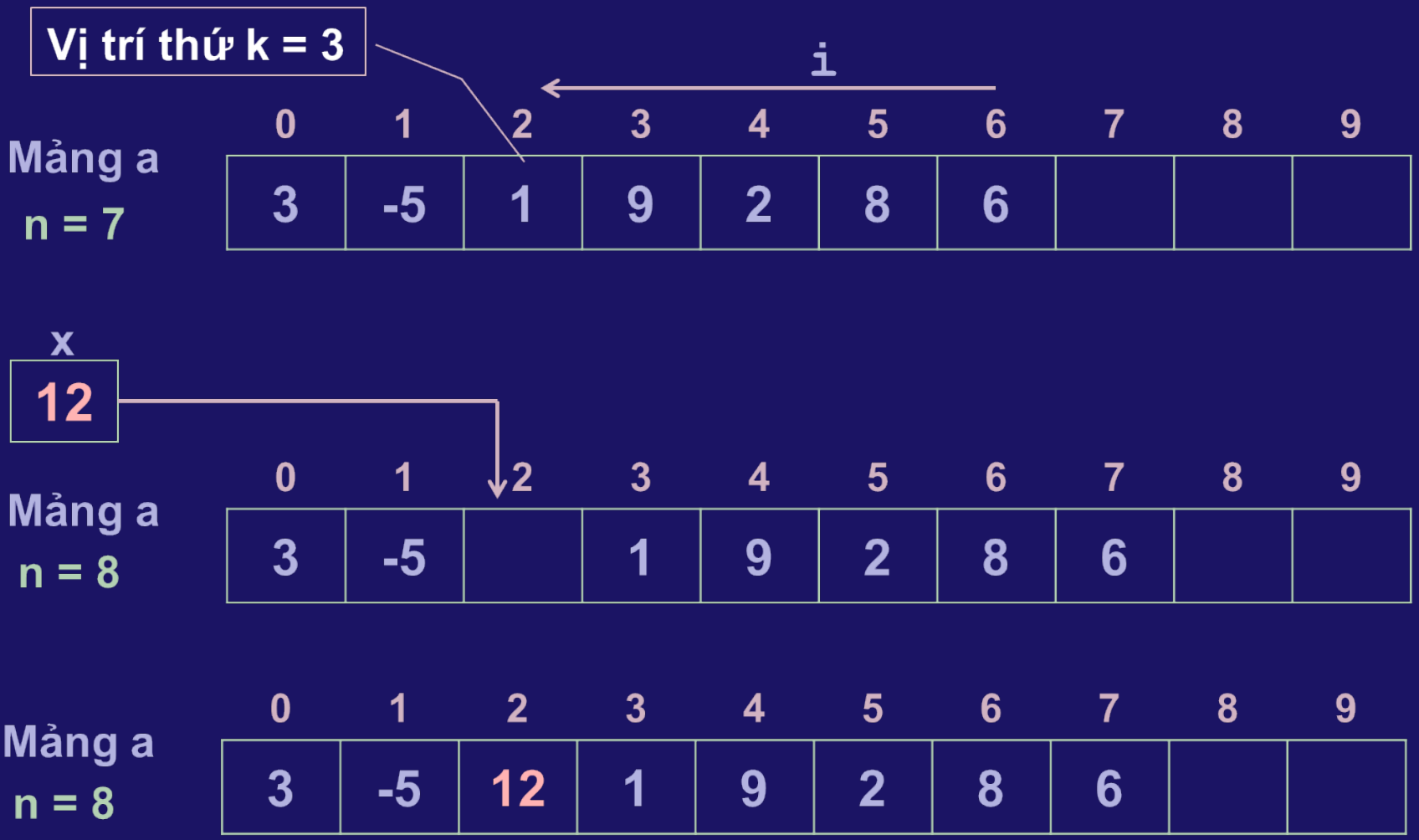
- Xóa phần tử thứ k = 3 trong mảng a đang có n phần tử dữ liệu.




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

4.1.3.4. Chèn một phần tử dữ liệu vào mảng

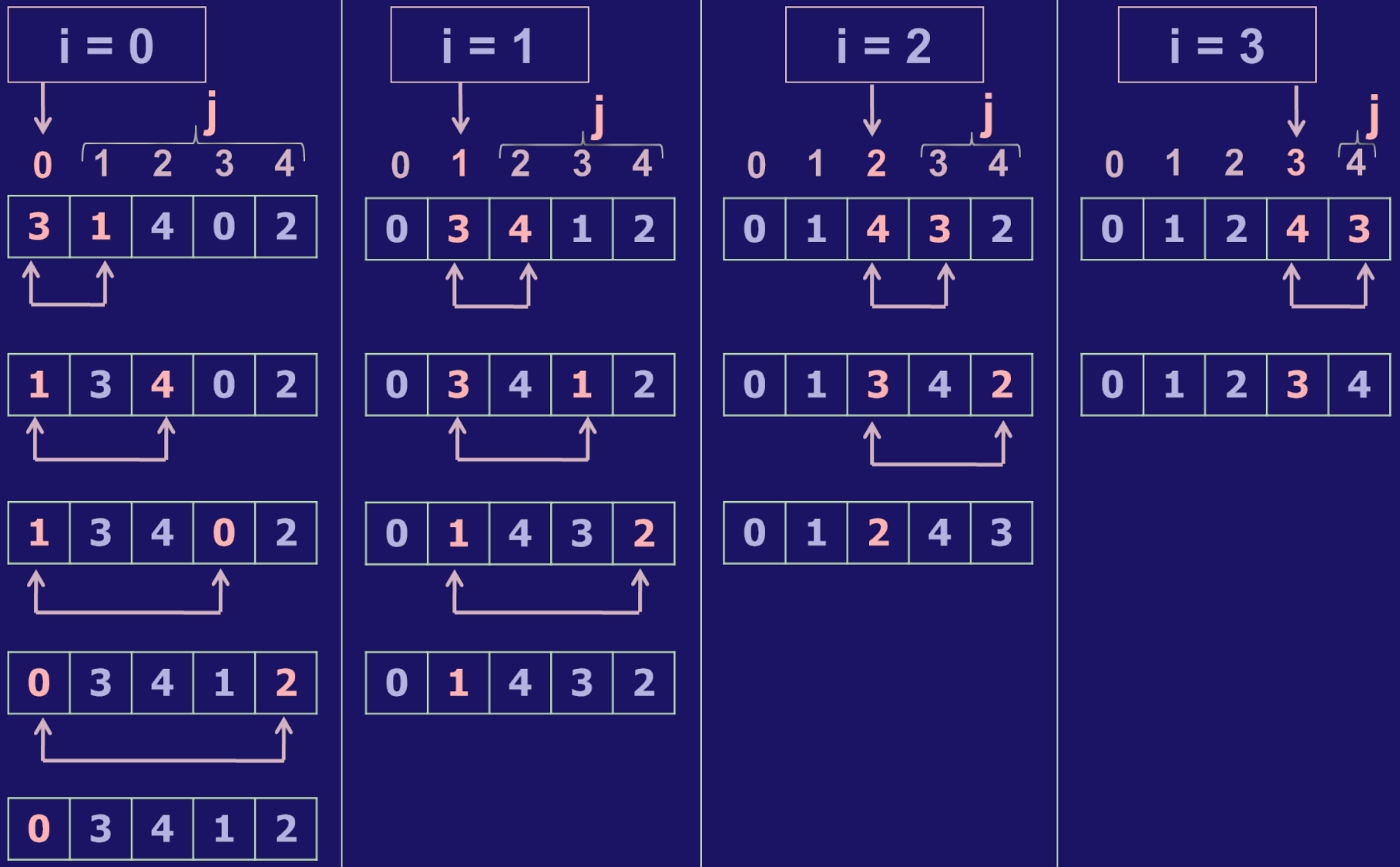
- Chèn phần tử x = 12 vào vị trí thứ k = 3 trong mảng a.



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

4.1.3.5. Sắp xếp dữ liệu trong mảng

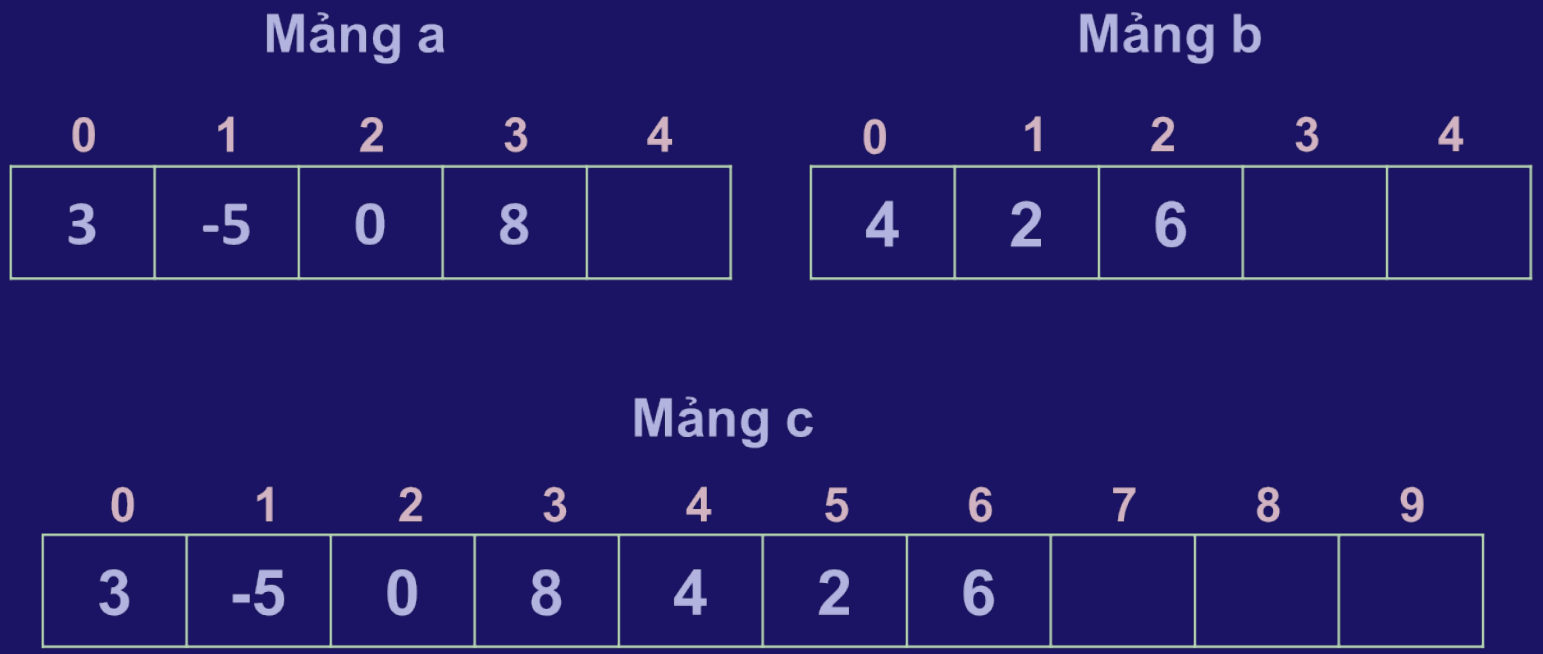


Sắp xếp dữ liệu trong mảng (tt)

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

4.1.3.6. Ghép mảng

- Ghép mảng a có 4 phần tử dữ liệu với mảng b có 3 phần tử dữ liệu được mảng c có 7 phần tử dữ liệu.



Ghép mảng (tt)

```
#define N 5
#define M 10
void ghepMang(int a[N], int m, int b[N],
              int n, int c[M], int &k) {
    k = 0;
    for (int i = 0; i < m; i++) {
        c[k] = a[i]; k++;
    }
    for (int i = 0; i < n; i++) {
        c[k] = b[i]; k++;
    }
}
```

4.1.3.7. Tách mảng

- Tách mảng a thành hai mảng b gồm các số dương và c là các số còn lại.

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

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

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

Tách mảng (tt)

```
#define M 10
void tachMang(int a[M], int m,
              int b[M], int &n,
              int c[M], int &k)
{
    n = k = 0;
    for (int i = 0; i < m; i++){
        if (a[i] > 0){
           b[n] = a[i]; n++;
        }
        else{
           c[k] = a[i]; k++;
        }
    }
}
```

4.1.4. Bài tập

- Cài đặt chương trình thực hiện các yêu cầu:
  - Nhập vào một mảng n số thực (1 <= n <= 30, n nhập từ bàn phím).
  - Hiển thị mảng lên màn hình.
  - Tìm và in ra màn hình giá trị nhỏ nhất.
  - Tính và in ra màn hình giá trị trung bình cộng của các số âm trong mảng.
  - Xóa phần tử thứ k trong mảng (k nhập từ bàn phím), in mảng vừa xóa lên màn hình.
  - Sắp xếp mảng theo chiều giảm dần, in mảng vừa sắp ra màn hình.