



IT001 - NHẬP MÔN LẬP TRÌNH

CHƯƠNG 7: MẢNG

Mảng (array) là một cấu trúc dữ liệu cơ bản trong ngôn ngữ lập trình C++. Nó cho phép lưu trữ một tập hợp các phần tử có cùng kiểu dữ liệu và được truy cập thông qua chỉ số. Mảng có kích thước cố định được xác định khi khai báo và không thể thay đổi trong khi chương trình đang chạy. Các kiến thức cơ bản về mảng một chiều, mảng hai chiều, mảng nhiều chiều và chuỗi sẽ được trình bày chi tiết trong chương này.

Khoa Khoa học Máy tính



NỘI DUNG

7.1 Đặt vấn đề

7.2 Khái niệm mảng và lợi ích của mảng

7.3 Các yếu tố xác định mảng

7.4 Mảng một chiều

7.5 Mảng hai chiều

7.6 Mảng nhiều chiều

7.7 Chuỗi C-String



NỘI DUNG

7.1 Đặt vấn đề

7.2 Khái niệm mảng và lợi ích của mảng

7.3 Các yếu tố xác định mảng

7.4 Mảng một chiều

7.5. Mảng hai chiều

7.6 Mảng nhiều chiều

7.7 Chuỗi C-String



7.1 Đặt vấn đề



1.1 Đặt vấn đề

- Chương trình cần lưu trữ 3 số thực
 - Khai báo 3 biến kiểu số thực: float a, b, c;
 - Chương trình cần lưu trữ 10 hoặc 100 hoặc 1000 số thực
 - Khai báo 10 hoặc 1000 hoặc n biến kiểu số thực?
 - Không thực hiện được
 - Cần có 1 kiểu dữ liệu mới để có thể lưu trữ dãy số thực này và truy xuất dễ dàng → **MẢNG**



Ví dụ:

```
#include<iostream>
using namespace std;

int main() {
    int a1, a2, a3, a4, a5, a6, a7, a8 , a9, a10;

    cin >> a1 >> a2 >> a3 >> a4 >> a5 >> a6
        >> a7 >> a8 >> a9 >> a10;

    if(a1 < 1989) cout << a1 << endl;
    if(a2 < 1989) cout << a2 << endl;
    if(a3 < 1989) cout << a3 << endl;
    if(a4 < 1989) cout << a4 << endl;
    if(a5 < 1989) cout << a5 << endl;
    if(a6 < 1989) cout << a6 << endl;
    if(a7 < 1989) cout << a7 << endl;
    if(a8 < 1989) cout << a8 << endl;
    if(a9 < 1989) cout << a9 << endl;
    if(a10 < 1989) cout << a10 << endl;
    return 0;
}
```

```
#include<iostream>
using namespace std;

int main() {
    int a[10];
    for(int i=0; i<10; i++)
        cin >> a[i];
    for(int i=0; i<10; i++)
        if(a[i]<1989)
            cout << a[i] << endl;
    return 0;
}
```



7.2 Khái niệm Mảng và lợi ích của mảng



Khái niệm Mảng

- Mảng biểu diễn một dãy các phần tử có **cùng kiểu** và mỗi phần tử trong mảng biểu diễn 1 giá trị.
- Kích thước mảng được **xác định ngay khi khai báo** và **không thay đổi**.
- Một kiểu dữ liệu có cấu trúc do người lập trình định nghĩa.
- Ngôn ngữ lập trình C++ luôn chỉ định một **khối nhớ liên tục** cho một biến kiểu mảng.
- C++ hỗ trợ mảng một chiều và mảng đa chiều (đơn giản nhất của mảng đa chiều là mảng hai chiều).
- Mảng một chiều các ký tự được dùng để biểu diễn chuỗi C-string.



Khái niệm mảng

- Mảng 1 chiều 10 giá trị số nguyên:

0x61fdf0 0x61fdf4 0x61fdf8 0x61fdfe 0x61fe00 0x61fe04 0x61fe08 0x61fe0c 0x61fe10 0x61fe14

0	9	3	2	0	9	1	0	6	0
---	---	---	---	---	---	---	---	---	---

- Mảng 1 chiều 5 ký tự: 0x61fdeb 0x61fdec 0x61fded 0x61fdee 0x61fdef

'h'	'e'	'l'	'l'	'o'
-----	-----	-----	-----	-----

- Mảng 2 chiều số thực 2 dòng 3 cột:

0.1 9.75 3.6 0x61fdb0 0x61fdb4 0x61fdb8 0x61fdbc 0x61fdc0 0x61fdc4

2.05	0.725	9.5	0.1	9.75	3.6	2.05	0.725	9.5
------	-------	-----	-----	------	-----	------	-------	-----

- Chuỗi "hello" là mảng 1 chiều gồm 6 ký tự:

0x61fde5 0x61fde6 0x61fde7 0x61fde8 0x61fde9 0x61fdea

'h'	'e'	'l'	'l'	'o'	'\0'
-----	-----	-----	-----	-----	------



Lợi ích của mảng

- **Truy cập dữ liệu nhanh chóng:** Truy cập phần tử dễ dàng thông qua chỉ số → truy cập và thay đổi giá trị
- **Quản lý dữ liệu đồng nhất:** Mảng chỉ chứa các phần tử có cùng kiểu dữ liệu
- **Đơn giản và dễ sử dụng:** Cú pháp của mảng tương đối đơn giản và dễ hiểu
- **Tích hợp với các hàm và thư viện:** C++ cung cấp nhiều hàm và thư viện hỗ trợ thao tác với mảng
- **Tính linh hoạt trong sử dụng:** Mảng có thể được sử dụng để triển khai nhiều cấu trúc dữ liệu khác nhau như hàng đợi (queue), ngăn xếp (stack)



7.3 Các yếu tố xác định mảng



7.3 Các yếu tố xác định mảng

- Các yếu tố xác định mảng gồm:

- **Tên mảng:**

- Tên gọi của mảng, được đặt theo quy tắc đặt tên biến
- Thông qua tên gọi kèm chỉ số có thể truy xuất đến từng phần tử của mảng

- **Kiểu mảng:** Là kiểu dữ liệu của các phần tử trong mảng

- **Số chiều:** Mảng 1 chiều, mảng 2 chiều hay mảng đa chiều

- **Kích thước:** Số lượng phần tử tối đa có thể lưu trữ theo mỗi chiều của mảng



7.3 Các yếu tố xác định mảng

- **Ví dụ:** Mảng a là mảng 1 chiều lưu trữ 10 số nguyên:

{0, 9, 3, 2, 0, 9, 1, 0, 6, 0}

0x61fdf0	0x61fdf4	0x61fdf8	0x61fdfc	0x61fe00	0x61fe04	0x61fe08	0x61fe0c	0x61fe10	0x61fe14
0	9	3	2	0	9	1	0	6	0

- **Tên mảng:** a
- **Kiểu mảng:** int
- **Số chiều:** 1 chiều
- **Kích thước:** 10 phần tử



7.3 Các yếu tố xác định mảng

- **Ví dụ:** Mảng str là mảng 1 chiều lưu trữ 5 ký tự: 'h', 'e', 'l', 'l', 'o'.
- **Tên mảng:** str
- **Kiểu mảng:** char
- **Số chiều:** mảng 1 chiều
- **Kích thước:** 5 phần tử

0x61fdeb	0x61fdec	0x61fded	0x61fdee	0x61fdef
'h'	'e'	'l'	'l'	'o'



7.3 Các yếu tố xác định mảng

- **Ví dụ:** Mảng b là mảng 2 chiều lưu trữ 2 dòng 3 cột các giá trị số thực như sau:

0.1	9.75	3.6
2.05	0.725	9.5

- **Tên mảng:** b
- **Kiểu mảng:** float
- **Số chiều:** mảng 2 chiều
- **Kích thước:** 2 dòng x 3 cột = 6 phần tử

0x61fdb0	0x61fdb4	0x61fdb8	0x61fdbc	0x61fdc0	0x61fdc4
0.1	9.75	3.6	2.05	0.725	9.5



7.4 Mảng 1 chiều



7.4 Mảng 1 chiều

7.4.1 Khai báo và khởi tạo mảng 1 chiều

7.4.2 Chỉ số mảng và truy xuất phần tử mảng

7.4.3 Lấy kích thước mảng dùng toán tử sizeof

7.4.4 Lấy địa chỉ các phần tử mảng

7.4.5 Phép Gán dữ liệu kiểu mảng

7.4.6 Truyền mảng cho hàm và lời gọi hàm

7.4.7 Một số tác vụ trên mảng 1 chiều



7.4 Mảng 1 chiều

7.4.1 Khai báo và khởi tạo mảng 1 chiều



Khai báo mảng 1 chiều

- Cú pháp:

```
<Kiểu_dữ_liệu> <Tên_biến_mảng>[<Số_phần_tử>];
```

- Trong đó:

- <Kiểu_dữ_liệu>: int, float, char, ...
- <Tên_biến_mảng>: Đặt tên theo quy tắc đặt tên/định danh
- <Số_phần_tử>:
 - Là số lượng tối đa các phần tử mảng có thể lưu trữ
 - Cần xác định ngay khi khai báo
 - Là một hằng số nguyên (literal, hoặc dùng `const` hoặc `define` để định nghĩa)



Khai báo mảng 1 chiều

- <Số_phần_tử_mảng> phải xác định cụ thể ngay lúc khai báo:

```
int a[100];
```

Kiểu dữ liệu: int

Tên biến mảng: a

Số phần tử mảng tối đa: 100 phần tử

- Sử dụng biến hằng (**const**) để định nghĩa số phần tử mảng:

```
const int MAXN = 100;  
int c[MAXN];
```

Kiểu dữ liệu: int

Tên biến mảng: c

Số phần tử mảng tối đa: 100 phần tử

- <Số_phần_tử_mảng> của mảng là biến sẽ không được chấp nhận:

```
int n1 = 10;  
int b[n1]; => Lỗi
```

- Sử dụng chỉ thị tiền xử lý **#define** để định nghĩa số phần tử mảng:

```
#define MAXN 100
```

```
float d[MAXN];
```

Kiểu dữ liệu: float

Tên biến mảng: d

Số phần tử mảng tối đa: 100 phần tử



Khai báo 1 chiều

- Cú pháp khai báo không tường minh:

```
typedef <kiểu_dữ_liệu_mảng> <Tên_kiểu_mảng>[<Số_phần_tử>];  
<Tên_kiểu_mảng> <Tên_bien_Mảng>;
```

- Ví dụ:

```
typedef int Mang1Chieu[10];  
  
Mang1Chieu m1, m2, m3;
```

```
// Khai báo trên tương đương với khai báo :  
  
int m1[10], m2[10], m3[10];
```



Khởi tạo mảng 1 chiều

- Khởi tạo giá trị cho mọi phần tử của mảng

`int A[4] = {10, 20, 30, 40};`

0	1	2	3
10	20	30	40

- Khởi tạo giá trị cho một số phần tử đầu mảng

`int B[4] = {10, 20};`

0	1	2	3
10	20	0	0

- Khởi tạo giá trị 0 cho mọi phần tử của mảng

`int C[4] = {};`

0	1	2	3
0	0	0	0

- Tự động xác định số lượng phần tử

`int E[] = {10, 20, 30, 40};`

0	1	2	3
10	20	30	40

`int F[] = {10, 20, 30, 40};`

0	1	2	3
10	20	30	40



7.4 Mảng 1 chiều

7.4.2 Chỉ số mảng và truy xuất phần tử mảng



7.4.2 Chỉ số mảng và truy xuất phần tử mảng

- Chỉ số mảng (vị trí trong mảng) là một giá trị số nguyên int.
- Chỉ số bắt đầu là 0 và không vượt quá số lượng phần tử tối đa trong mảng.
- Số lượng các chỉ số mảng = số lượng phần tử tối đa trong mảng.
- Ví dụ:

`int A[5];`

0	1	2	3	4
99	17	50	43	72

Tên mảng: A

Kiểu dữ liệu của từng phần tử trong mảng: int

Số phần tử tối đa trong mảng: 5 phần tử

Các chỉ số được đánh số: 0 → 4 (0, 1, 2, 3, 4)



7.4.2 Chỉ số mảng và truy xuất phần tử mảng

- Truy xuất phần tử mảng thông qua chỉ số:

```
<Tên biến mảng>[<Chỉ số mảng>];
```

- Các phần tử là 1 dãy liên tục có chỉ số từ 0 đến <Số_phần_tử_mảng>-1
- Ví dụ:

<code>int A[5];</code>	0	1	2	3	4
	99	17	50	43	72

Các truy xuất hợp lệ: A[0], A[1], A[2], A[3], A[4]

Các truy xuất không hợp lệ: A[-1], A[5], A[6]

Giá trị các phần tử mảng A[0]=99, A[1]=17, A[2]=50,

A[3]=43, A[4]=72



7.4.2 Chỉ số mảng và truy xuất phần tử mảng

- Ví dụ:

```
#include<iostream>
using namespace std;

int main() {
    int a[5]={10, 20, 30, 40, 50};
    for(int i=0; i<5; i++)
        cout << a[i] << " ";
    cout << endl;
    return 0;
}
```

0x61fe00	0x61fe04	0x61fe08	0x61fe0c	0x61fe10
10	20	30	40	50

Kết quả thực thi:

10 20 30 40 50



7.4 Mảng 1 chiều

7.4.3 Lấy kích thước mảng dùng toán tử sizeof



7.4.3 Lấy kích thước mảng dùng toán tử sizeof

- Cú pháp: Cho mảng 1 chiều A có n phần tử
 - `sizeof(A)`: kích thước toàn mảng
 - `sizeof(A[i])`: lấy kích thước phần tử thứ i ($0 \leq i < n$) trong mảng
- Ví dụ:

```
#include<iostream>
using namespace std;
int main() {
    int A[] = { 1, 2, 3, 4, 5, 6 };
    cout << "Kich thuoc mang: " << sizeof(A) << " bytes." << endl;
    cout << "Kich thuoc phan tu A[0]: " << sizeof(A[0]) << " byte." << endl;
    cout << "Mang co: " << sizeof(A)/sizeof(int) << " phan tu.";
    return 0;
}
```

Kết quả thực thi:

Kich thuoc mang: 24 bytes.
Kich thuoc phan tu A[0]: 4 byte.
Mang co: 6 phan tu.



7.4 Mảng 1 chiều

7.4.4 Lấy địa chỉ các phần tử mảng



7.4.4 Lấy địa chỉ các phần tử mảng

- Cú pháp:

& <Tên biến mảng>[<Chỉ số mảng>];

- Ví dụ:

```
int A[5];
```

0	1	2	3	4
99	17	50	43	72
0x14	0x18	0x22	0x26	0x30

Địa chỉ của mảng: &A

Địa chỉ các phần tử mảng:

Địa chỉ phần tử thứ 0: &A[0] (&A[0] = &A)

Địa chỉ phần tử thứ 1: &A[1]

Địa chỉ phần tử thứ 2: &A[2]

Địa chỉ phần tử thứ 3: &A[3]

Địa chỉ phần tử thứ 4: &A[4]

Lưu ý: những giá trị 0x14, 0x18, 0x22, 0x26, 0x30 là các địa chỉ giả định



7.4.4 Lấy địa chỉ các phần tử mảng

- Ví dụ:

```
#include<iostream>
using namespace std;

int main() {
    int a[5]={10, 20, 30, 40, 50};
    for(int i=0; i<5; i++)
        cout << a[i] << " ";
    cout << endl;
    cout << a << endl;
    for(int i=0; i<5; i++)
        cout << &a[i] << " ";
    return 0;
}
```

0x61fe00	0x61fe04	0x61fe08	0x61fe0c	0x61fe10
10	20	30	40	50

Kết quả thực thi:

```
10 20 30 40 50
0x61fe00
0x61fe00 0x61fe04 0x61fe08 0x61fe0c 0x61fe10
```



7.4 Mảng 1 chiều

7.4.5 Phép Gán dữ liệu kiểu mảng



7.4.5 Phép Gán dữ liệu kiểu mảng

- Không được sử dụng phép gán thông thường mà phải gán trực tiếp giữa các phần tử tương ứng:

```
<Biến_mảng_đích> = <biến_mảng_nguồn>; → sai
```

```
<Tên_biến_mảng>[<Chỉ_số_thứ_i>] = <Giá_trị>; → đúng
```

- Ví dụ:

```
#define MAX 3
typedef int MangSo[MAX];
MangSo a = {1, 2, 3}, b;
b = a; // Sai
for (int i = 0; i < 3; i++)
    b[i] = a[i];
```



Ví dụ:

```
#include<iostream>
using namespace std;
#define MAX 3
int main() {
    typedef int MangSo[MAX];
    MangSo a = {1, 2, 3}, b; // hay: int a[MAX], b[MAX];
    for (int i = 0; i < 3; i++)
        b[i] = a[i];

    cout << "&a= " << &a << " " << endl;
    for (int i = 0; i < 3; i++)
        cout << "&a[" << i << "]=" << &a[i] << " " << endl;

    cout << "&b= " << &b << " " << endl;
    for (int i = 0; i < 3; i++)
        cout << "&b[" << i << "]=" << &b[i] << " " << endl;
}
```

0x61fe08 0x61fe0c 0x61fe10

1	2	3
---	---	---

a

0x61fdfc 0x61fe00 0x61fe04

1	2	3
---	---	---

b

Kết quả thực thi:

&a= 0x61fe08

&a[0]= 0x61fe08

&a[1]= 0x61fe0c

&a[2]= 0x61fe10

&b= 0x61fdfc

&b[0]= 0x61fdfc

&b[1]= 0x61fe00

&b[2]= 0x61fe04



Bài tập: Hãy cho biết các khai báo nào sau sai?

- Khai báo không chỉ rõ số lượng phần tử

```
int a[];
```

```
int a[100];
```

- Số lượng phần tử liên quan đến biến hoặc hằng

```
int n1 = 10; int a[n1];
```

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

```
const int n2 = 10; int a[n2];
```

- Khởi tạo cách biệt với khai báo

```
int a[4]; a = {2912, 1706, 1506, 1904};
```

```
int a[4] = {2912, 1706, 1506, 1904};
```

- Chỉ số mảng không hợp lệ

```
a[1] = 1; (truy xuất trong mảng có nhiều hơn 1 phần tử)
```

```
a[-1] = 1;
```



Bài tập: Hãy cho biết các khai báo nào sau sai?

- Khai báo không chỉ rõ số lượng phần tử

`int a[]; => LỖI`

`int a[100]; => OK`

- Số lượng phần tử liên quan đến biến hoặc hằng

`int n1 = 10; int a[n1]; => LỖI`

`int a[10]; => OK`

`const int n2 = 10; int a[n2]; => OK`

- Khởi tạo cách biệt với khai báo

`int a[4]; a = {2912, 1706, 1506, 1904}; => LỖI`

`int a[4] = {2912, 1706, 1506, 1904}; => OK`

- Chỉ số mảng không hợp lệ

`a[1] = 1; (truy xuất trong mảng có nhiều hơn 1 phần tử) => OK`

`a[-1] = 1; => LỖI`



7.4 Mảng 1 chiều

7.4.6 Truyền mảng cho hàm và lời gọi hàm



7.4.6 Truyền mảng cho hàm và lời gọi hàm

- Cú pháp khai báo tham biến mảng:

```
<Giá_Trả_Về> <Tên_Hàm> (<Kiểu_Dữ liệu> <Tên_Mảng>[<Số_phần_tử>],  
                                <Tham số khác>, ...);
```

- Mảng có thể thay đổi nội dung sau khi thực hiện hàm.
- Có thể bỏ số lượng phần tử hoặc sử dụng con trỏ.
- Ví dụ:

```
void NhapMang(int A[MAXN], int n);  
  
void NhapMang(int A[], int n);  
  
void NhapMang(int *A, int n);
```



7.4.6 Truyền mảng cho hàm và lời gọi hàm

- Ví dụ:

```
void NhapMang(int A[MAXN], int n);
```

```
void NhapMang(int A[], int n);
```

```
void NhapMang(int *A, int n);
```

Trong đó:

- Tên hàm: NhapMang
- Tham số: mảng số nguyên A và số lượng phần tử mảng n
- Giá trị trả về: void



7.4.6 Truyền mảng cho hàm và lời gọi hàm

- Ví dụ:

```
#include <iostream>
#define MAXN 100
void NhapMang(int*, int &);
void XuatMang(int[], const int &);
int TinhTong(int A[MAXN], int N);
int main() {
    int a[MAXN], n, S;
    NhapMang(a, n);
    XuatMang(a, n);
    S = TinhTong(a, n);
    std::cout << std::endl
    std::cout << "Tong cac phan tu trong mang: " << S;
    return 0;
}
```



7.4.6 Truyền mảng cho hàm và lời gọi hàm

- Ví dụ:

```
void NhapMang(int *A, int &N) {  
    std::cout << "Nhập N (0 ≤ N ≤ 100): ";  
    std::cin >> N;  
    for (int i = 0; i < N; i++) {  
        std::cout << "a[" << i << "] = ";  
        std::cin >> A[i];  
    }  
}
```

Kết quả thực thi:

Nhập N (0 ≤ N ≤ 100): 3

a[0]= 5

a[1]= 4

a[2]= 9

Mảng gồm: 5, 4, 9.

Tổng các phần tử trong mảng: 18

```
void XuatMang(int A[], const int &N)  
{  
    std::cout << "Mảng gồm: ";  
    for (int i = 0; i < N - 1; i++)  
        std::cout << A[i] << ", ";  
    std::cout << A[N - 1] << ". ";  
}
```

```
int TinhTong(int A[MAXN], int N) {  
    int S = 0;  
    for (int i = 0; i < N; i++)  
        S += A[i];  
    return S;  
}
```



7.4 Mảng 1 chiều

7.4.7 Một số tác vụ trên mảng 1 chiều



7.4.7 Một số tác vụ trên mảng 1 chiều

1. Nhập mảng
2. Xuất mảng
3. Tìm kiếm một phần tử trong mảng
4. Tìm giá trị nhỏ nhất/lớn nhất của mảng
5. Kiểm tra tính chất của mảng
6. Đếm số lượng các phần tử có giá trị chẵn trong mảng
7. Tách mảng / Gộp mảng
8. Sắp xếp mảng giảm dần/tăng dần
9. Thêm/Xóa/Sửa một phần tử vào mảng



1. Nhập mảng

- Yêu cầu: Cho phép nhập mảng a, số lượng phần tử n

```
void NhapMang(int A[], int N) {  
    for (int i = 0; i < N; i++) {  
        std::cout << "a[" << i << "] = ";  
        std::cin >> A[i];  
    }  
}
```



2. Xuất mảng

- **Yêu cầu:** Cho trước mảng a, số lượng phần tử n. Hãy xuất nội dung mảng a ra màn hình.

```
void XuatMang(int A[MAXN], int N) {  
    for (int i = 0; i < N; i++)  
        std::cout << A[i];  
}
```



3. Tìm kiếm 1 phần tử trong mảng

- **Yêu cầu:** Tìm xem phần tử x có nằm trong mảng a kích thước n hay không? Nếu có thì xuất ra màn hình vị trí đầu tiên tìm thấy được.

```
int TimKiem(int *a, int n, int x) {  
    for (int vt = 0; vt < n; vt++)  
        if (a[vt] == x) return vt;  
    return -1;  
}
```



4. Tìm giá trị nhỏ nhất/lớn nhất của mảng

```
int Max(int a[], int n) {  
    int Max = a[0];  
    for (int i = 1; i < n; i++)  
        if (Max < a[i]) // áp dụng kỹ thuật lính canh  
            Max = a[i];  
    return Max;  
}  
  
int Min(int a[], int n) {  
    int Min = a[0];  
    for (int i = 1; i < n; i++)  
        // áp dụng kỹ thuật lính canh  
        if (Min > a[i]) Min = a[i];  
    return Min;  
}
```



5. Kiểm tra tính chất của mảng

- **Yêu cầu**
 - Cho trước mảng a, số lượng phần tử n. Mảng a có phải là mảng toàn các số nguyên tố hay không?
- **Ý tưởng**
 - Ý tưởng 1: Đếm số lượng số nguyên tố của mảng. Nếu số lượng này bằng đúng n thì mảng toàn nguyên tố.
 - Ý tưởng 2: Đếm số lượng số không phải nguyên tố của mảng. Nếu số lượng này bằng 0 thì mảng toàn nguyên tố.
 - Ý tưởng 3: Tìm xem có phần tử nào không phải số nguyên tố không. Nếu có thì mảng không toàn số nguyên tố.



5. Kiểm tra tính chất của mảng

// Hàm kiểm tra SNT

```
int LaSNT(int n) {
```

```
    int flag = 0;
```

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

```
        if (n % i == 0) // áp dụng kỹ thuật cờ hiệu
```

```
            flag = 1;
```

```
    if (flag == 0) return 1; // SNT
```

```
    return 0; // Không phải SNT
```

```
}
```



5. Kiểm tra tính chất của mảng

```
int KiemTra_YT1(int a[], int n) {  
    int dem = 0;  
    for (int i = 0; i < n; i++)  
        if (LaSNT(a[i]) == 1)  
            dem++;  
    if (dem == n)  
        return 1;  
    return 0;  
}
```

```
int KiemTra_YT3(int a[], int n) {  
    for (int i = 0; i < n; i++)  
        if (LaSNT(a[i]) == 0)  
            return 0;  
    return 1;  
}
```

```
int KiemTra_YT2(int a[], int n){  
    int dem = 0;  
    for (int i = 0; i < n; i++)  
        if (LaSNT(a[i]) == 0)  
            dem++;  
    if (dem == 0)  
        return 1;  
    return 0;  
}
```



6. Đếm số lượng các phần tử chẵn trong mảng

```
#include <stdio.h>
void NhapMang(int A[], int &N);
void XuatMang(int A[], int N); void
DemSoChan(int A[], int N);

int main() {
    int a[100], n, count;

    NhapMang(a, n);
    XuatMang(a, n);

    count = DemSoChan(a, n);
    printf("So phan tu chan la %d", count);
}
```

```
int DemSoChan(int A[], int N) {
    int c = 0;
    for (int i = 0; i < n; i++)
        if (A[i] % 2 == 0)
            c++;
    return c;
}
```



7. Tách mảng / Gộp mảng

- Yêu cầu
 - Cho trước mảng a, số lượng phần tử na và mảng b số lượng phần tử nb. Gộp 2 mảng trên theo thứ tự đó thành mảng c, số lượng phần tử nc.
- Ý tưởng
 - Chuyển các phần tử của mảng a sang mảng c
 $\Rightarrow nc = na$
 - Tiếp tục đưa các phần tử của mảng b sang mảng c
 $\Rightarrow nc = nc + nb$



7. Tách mảng / Gộp mảng

```
void GopMang(int a[], int na, int b[], int nb, int c[], int &nc) {  
    nc = 0;  
  
    for (int i = 0; i < na; i++) {  
        c[nc] = a[i];  
        nc++; // c[nc++] = a[i];  
    }  
  
    for (int i = 0; i < nb; i++) {  
        c[nc] = b[i];  
        nc++; // c[nc++] = b[i];  
    }  
}
```



7. Tách mảng / Gộp mảng

- Yêu cầu
 - Cho trước mảng a, số lượng phần tử na. Tách mảng a thành 2 mảng b (chứa số nguyên tố) và mảng c (các số còn lại).
- Ý tưởng
 - Cách 1: viết 1 hàm tách các số nguyên tố từ mảng a sang mảng b và 1 hàm tách các số không phải nguyên tố từ mảng a sang mảng c.
 - Cách 2: Duyệt từng phần tử của mảng a, nếu đó là số nguyên tố thì đưa vào mảng b, ngược lại đưa vào mảng c.



7. Tách mảng / Gộp mảng

```
void TachSNT2(int a[], int na, int b[], int &nb, int c[], int &nc) {  
    nb = 0;  
    nc = 0;  
  
    for (int i = 0; i < na; i++)  
        if (LaSNT(a[i]) == 1) {  
            b[nb] = a[i];  
            nb++;  
        }  
        else {  
            c[nc] = a[i];  
            nc++;  
        }  
}
```



8. Sắp xếp mảng giảm dần/tăng dần

- Yêu cầu
 - Cho trước mảng a kích thước n . Hãy sắp xếp mảng a đó sao cho các phần tử có giá trị **tăng dần**.
- Ý tưởng
 - Sử dụng 2 biến i và j để so sánh tất cả cặp phần tử với nhau và hoán vị các cặp **nghịch thế** (sai thứ tự).



8. Sắp xếp mảng giảm dần/tăng dần

```
void HoanVi(int &a, int &b) {  
    int temp = a;  
    a = b;  
    b = temp;  
}  
  
void SapXepTang_InterchangeSort(int a[], int n) {  
    for (int i = 0; i < n-1; i++)  
        for (int j = i + 1; j < n; j++)  
            if (a[i] > a[j])  
                HoanVi(a[i], a[j]);  
}
```



9. Thêm/Xóa/Sửa một phần tử vào mảng

- Yêu cầu
 - Thêm phần tử x vào mảng a kích thước n tại vị trí k .
- Ý tưởng
 - “Đẩy” các phần tử bắt đầu tại vị trí k sang phải 1 vị trí.
 - Đưa x vào vị trí k trong mảng.
 - Tăng n lên 1 đơn vị.



9. Thêm/Xóa/Sửa một phần tử vào mảng

```
bool Them(int a[], int &n, int k, int x) {  
    if (k >= 0 && k <= n) {  
        for (int i = n; i > k; i--)  
            a[i] = a[i - 1];  
        a[k] = x;  
        n++;  
        return 1;  
    }  
    return 0;  
}
```



9. Thêm/Xóa/Sửa một phần tử vào mảng

- Yêu cầu
 - Xóa một phần tử trong mảng a kích thước n tại vị trí **k**
- Ý tưởng
 - “Kéo” các phần tử bên phải vị trí **k** sang trái 1 vị trí.
 - Giảm **n** xuống **1 đơn vị**.



9. Thêm/Xóa/Sửa một phần tử vào mảng

```
bool Xoa(int a[], int &n, int k) {  
    if (k >= 0 && k < n) {  
        for (int i = k; i < n - 1; i++)  
            a[i] = a[i + 1];  
  
        n--;  
  
        return 1;  
    }  
  
    return 0;  
}
```



Bài tập

- Câu 1: Viết chương trình nhập vào một dãy tăng dần, không cần sắp xếp. Nếu nhập sai yêu cầu sẽ phải nhập lại và xuất các số nguyên tố có trong mảng.
- Câu 2: Kiểm tra mảng có đối xứng hay không?
- Câu 3: Liệt kê các giá trị xuất hiện trong mảng đúng 1 lần.
- Câu 4: Tìm vị trí của phần tử có giá trị âm lớn nhất trong mảng số nguyên.
- Câu 5: Viết hàm xóa phần tử có chỉ số k trong mảng số nguyên a có n phần tử. Nếu giá trị của $k < 0$ hoặc $k = n$ thì không xóa và hàm trả về giá trị 0. Ngược lại ta xóa giá trị phần tử $a[k]$ và hàm trả về giá trị 1.



Chúc các em học tốt !

