

# Chương 01.6: Mảng và xâu ký tự

I. Mảng

II. Xâu ký tự

III. Bài tập chương 6

# I. Mảng

1. Khái niệm về kiểu mảng
2. Khai báo biến mảng một chiều
3. Các phần tử của mảng một chiều
4. Truy nhập các phần tử của mảng một chiều
5. Khởi tạo mảng một chiều
6. Mảng nhiều chiều
7. Chú ý về chỉ số của phần tử mảng
8. Vào/ra với biến mảng

# 1. Khái niệm về kiểu mảng

- ✧ Mảng là một nhóm các biến nằm cạnh nhau có cùng kiểu, cùng tên. Mỗi biến được gọi là một phần tử. Các phần tử của mảng được truy nhập trực tiếp thông qua tên biến mảng và chỉ số.
- ✧ Số phần tử của mảng được xác định ngay từ khi định nghĩa ra mảng. Đây là điểm hạn chế của mảng bởi vì nếu không dùng hết các biến của mảng sẽ gây lãng phí bộ nhớ.

## 2. Khai báo biến mảng một chiều

- ✧ Khai báo biến mảng là xác định tên biến mảng, kiểu phần tử, số chiều và kích thước mỗi chiều.
- ✧ Cú pháp khai báo biến mảng một chiều:

Kiểu\_phần\_tử Tên\_biến\_mảng[Kích\_thước];

trong đó kích thước là số phần tử của mảng, phải cho dưới dạng hằng hoặc biểu thức hằng. Kiểu phần tử có thể là bất kỳ kiểu nào.

*Ví dụ:* int a[5];

Ví dụ này định nghĩa một biến mảng có tên là a, kiểu phần tử là int, số chiều là một và kích thước (số phần tử cực đại của mảng) là 5.

### 3. Các phần tử của mảng một chiều

- ✧ Các phần tử của mảng được đánh số. Các số này gọi là chỉ số. Phần tử đầu tiên có chỉ số là 0, phần tử thứ 2 có chỉ số là 1,... Mảng có kích thước  $n$  thì phần tử cuối cùng có chỉ số  $n-1$ .
- ✧ *Ví dụ:* nếu ta định nghĩa một biến mảng  
`int a[5];`  
thì ta được một biến mảng tên là `a` có 5 phần tử, phần tử đầu tiên có chỉ số là 0, phần tử thứ 5 có chỉ số là 4.

## 4. Truy nhập các phần tử của mảng một chiều

✧ Mỗi phần tử của mảng có thể truy nhập trực tiếp thông qua tên biến mảng và chỉ số của nó đặt trong ngoặc vuông []. Chỉ số của phần tử có thể cho dưới dạng hằng hoặc biểu thức.

✧ *Ví dụ:* 5 phần tử của mảng a ở ví dụ trên có tên là a[0], a[1],... Ta có thể dùng các lệnh sau:

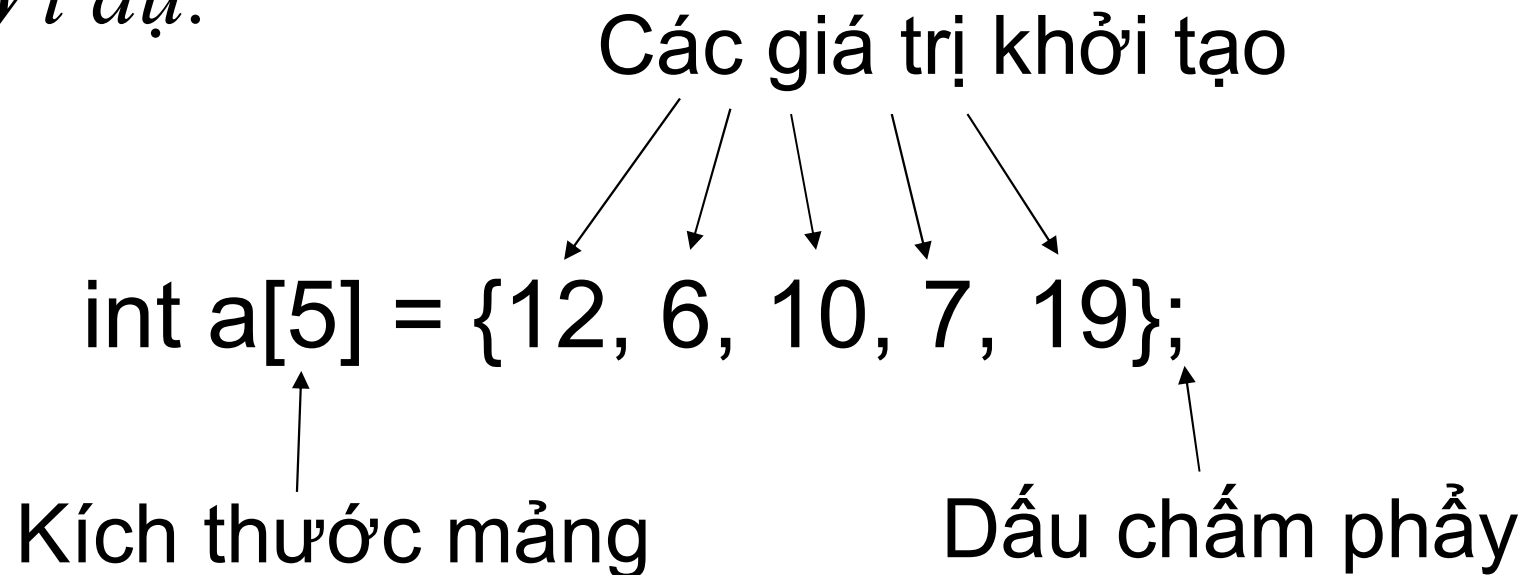
```
a[0]=100; cout<<a[1];
```

```
for(int i=0;i<5;++i) cin>>a[i];
```

## 5. Khởi tạo mảng một chiều

✧ Ta có thể khởi tạo giá trị cho các phần tử của mảng ngay khi định nghĩa bằng cách liệt kê các giá trị khởi tạo đặt trong ngoặc {}.

✧ Ví dụ:



## 5. Khởi tạo mảng một chiều (tiếp)

- ✧ Nếu số giá trị khởi tạo ít hơn kích thước mảng thì các phần tử còn lại sẽ được khởi tạo bằng 0. Nếu số giá trị khởi tạo lớn hơn kích thước mảng thì trình biên dịch sẽ báo lỗi.

*Ví dụ:* `int a[3] = {6,8}; //a[0]=6, a[1]=8, a[2]=0`

`int a[2] = {8, 6, 9}; //Báo lỗi`

- ✧ Với những mảng được khởi tạo có thể không cần xác định kích thước mảng. Khi đó trình biên dịch sẽ đếm số giá trị khởi tạo và dùng số đó làm kích thước mảng. *Ví dụ:*

`int a[] = {3, 5, 8}; //sẽ được mảng có kích thước là 3`



## 6. Mảng nhiều chiều

- ✧ Mảng một chiều là mảng mà các phần tử của nó được truy nhập qua một chỉ số. Mảng nhiều chiều là mảng mà các phần tử được truy nhập qua nhiều chỉ số.
- ✧ C++ cho phép khai báo các mảng nhiều chiều với kích thước mỗi chiều có thể khác nhau. Cú pháp chung như sau:

Kiểu Tên\_biến\_mảng[Kích thước chiều 1][Kích thước chiều 2]...;

- ✧ Ví dụ:

```
int a[4][3];
```

Lưu ý là mỗi chiều phải được bao bởi cặp ngoặc []

## 6. Mảng nhiều chiều (tiếp)

- ✧ Để truy nhập phần tử của mảng  $m$  chiều thì ta phải dùng  $m$  chỉ số. Chỉ số của mỗi chiều có giá trị từ 0 đến kích thước của chiều đó trừ đi 1. Cú pháp chung như sau:

Tên\_biến\_mảng[chỉ số chiều 1][Chỉ số chiều 2]...

- ✧ Mảng 2 chiều có thể xem như là mảng một chiều có các phần tử là một mảng một chiều.
- ✧ Ta cũng có thể khởi tạo giá trị cho các phần tử của mảng nhiều chiều ngay khi định nghĩa. Ví dụ:  
$$\text{int } a[2][3] = \{\{5, 7, 9\}, \{3, 6, 7\}\};$$

## 7. Chú ý về chỉ số của phần tử mảng

✧ Trình biên dịch C++ sẽ không báo lỗi khi chỉ số dùng để truy nhập phần tử của mảng nằm ngoài khoảng cho phép, tức là nhỏ hơn 0 hoặc lớn hơn kích thước mảng trừ 1. Điều này rất nguy hiểm bởi vì nếu ta ghi dữ liệu vào phần tử mảng với chỉ số nằm ngoài khoảng cho phép thì có thể ghi đè lên dữ liệu của các chương trình khác đang chạy hoặc chính chương trình của ta.

## 8. Vào/ra với biến mảng

- ✧ Không dùng được lệnh `cout` và `cin` với cả biến mảng.
- ✧ Chỉ dùng được `cout` và `cin` với từng phần tử của mảng. Ví dụ:

```
int a[5];
for(int i=0;i<5;++i)
    {cout<<"Nhập vào phần tử thứ "<<i+1<<": ";
      cin>>a[i];
    }
for(int i=0;i<5;++i) cout<<a[i]<<' ';
```

## II. Xâu ký tự

1. Khái niệm về kiểu xâu ký tự
2. Khai báo biến xâu ký tự
3. Khởi tạo biến xâu ký tự
4. Vào/ra với biến xâu
5. Các hàm chuẩn xử lý xâu ký tự
6. Mảng xâu ký tự

# 1. Khái niệm về kiểu xâu ký tự

- ✧ Xâu ký tự là một dãy ký tự có ký tự cuối cùng là ký tự rỗng. Ký tự rỗng có giá trị số là 0 và viết là '\0'.
- ✧ Xâu ký tự được C++ lưu trữ như một mảng ký tự, nó cho phép truy nhập vào từng ký tự của xâu như truy nhập vào từng phần tử của mảng. Tuy nhiên, trong một số trường hợp C++ xem xâu ký tự như những kiểu dữ liệu cơ bản. Ví dụ, có thể nhập vào và đưa ra cả biến xâu bằng lệnh cout và cin.

## 2. Khai báo biến xâu ký tự

- ✧ Khai báo biến xâu ký tự là xác định tên biến xâu và số ký tự cực đại có thể chứa trong biến xâu.
- ✧ Cú pháp khai báo biến xâu ký tự giống cú pháp khai báo biến mảng một chiều:  
char Tên\_biến\_xâu[Kích thước];  
trong đó số ký tự cực đại cho dưới dạng hằng hoặc biểu thức hằng.
- ✧ Biến xâu có thể chứa các xâu ký tự có độ dài khác nhau nhưng không vượt quá kích thước biến xâu -1.

### 3. Khởi tạo biến xâu

✧ Khi định nghĩa biến xâu ta có thể khởi tạo cho nó. Dưới đây là 2 cách khởi tạo:

- Khởi tạo như biến mảng:

```
char str[6] = {'D', 'H', 'N', 'N', 'I', '\0'};
```

- Khởi tạo bằng hằng xâu:

```
char str[6] = "DHNNI";
```

Hằng xâu là một dãy ký tự đặt giữa 2 dấu phẩy kép. Khi viết hằng xâu ta không viết ký tự '\0', ký tự này sẽ được trình biên dịch thêm vào. Hằng xâu rỗng là hằng xâu không có ký tự nào "".



### 3. Khởi tạo biến xâu (tiếp)

- ✧ Lưu ý là khi khởi tạo cho biến xâu bằng hằng xâu thì số ký tự cực đại của biến xâu phải lớn hơn số ký tự của hằng xâu ít nhất là 1, bởi vì trình biên dịch sẽ đưa thêm vào biến xâu một ký tự rỗng. Ví dụ:

```
char str[5] = "DHNNI"; //Sai
```

```
char str[6] = "DHNNI"; //Đúng
```

- ✧ Cũng giống như biến mảng, khi khởi tạo cho biến xâu thì có thể không cần xác định số ký tự cực đại, khi đó trình biên dịch sẽ xác định số ký tự cực đại bằng số ký tự của hằng xâu cộng thêm 1. Ví dụ:

```
char str[] = "DHNNI";
```

## 4. Vào/ra với biến xâu

✧ Có thể dùng lệnh `cout` và `cin` với cả biến xâu. Ví dụ:

```
char str[11];
```

```
cin>>str; cout<<str;
```

✧ **Lưu ý:** Nếu dùng `cin` để nhập vào xâu ký tự thì không nhập được các xâu có khoảng cách vì khi gặp khoảng trắng `cin` sẽ kết thúc.

Để khắc phục nhược điểm trên ta dùng hàm thành viên của `cin` là `get` để lấy vào các xâu có cả khoảng cách:

*(xem tiếp trang sau)*

## 4. Vào/ra với biến xâu (tiếp)

**cin.get(Biến\_xâu, Kích\_thước\_biến\_xâu);**

Ví dụ: `char str[11]; cin.get(str, sizeof(str));`

`cin.get(str, sizeof(str));`

- ✧ **Thận trọng:** Các lệnh `cin` sau khi kết thúc vẫn để ký tự '\n' trong bộ đệm bàn phím. Trong khi đó ký tự '\n' lại làm hàm thành viên `cin.get()` kết thúc, bởi vậy nếu trước hàm thành viên `cin.get()` có lệnh `cin` thì hàm thành viên `cin.get()` sẽ không lấy được ký tự nào. Để khắc phục nhược điểm này, ta dùng hàm thành viên `cin.ignore()` để huỷ các ký tự '\n' trước khi dùng `cin.get()`. Ví dụ:

`cin>>a;`

`scanf(" "); cin.get(str, 11);`

## 5. Các hàm chuẩn xử lý chuỗi ký tự

- ✧ C++ có một thư viện hàm làm việc với chuỗi ký tự là `string.lib`. Muốn sử dụng các hàm này ta phải khai báo sử dụng:

```
#include<string.h>
```

- ✧ Hàm lấy độ dài của chuỗi: `strlen(s)` cho độ dài của chuỗi `s` (không tính ký tự `'\0'`)
- ✧ Hàm copy chuỗi: `strcpy(s1, s2)` copy chuỗi `s2` vào biến chuỗi `s1`, `s2` có thể là hằng chuỗi hoặc biến chuỗi.

## 5. Các hàm chuẩn xử lý chuỗi ký tự (tiếp)

- ✧ Hàm nối chuỗi: `strcat(s1,s2)` nối chuỗi `s2` vào cuối biến chuỗi `s1`, `s2` có thể là hằng chuỗi hoặc biến chuỗi, biến chuỗi `s1` phải có số ký tự cực đại đủ chứa các ký tự `s2` khi thêm vào.
- ✧ Hàm so sánh chuỗi: `strcmp(s1,s2)` so sánh hai chuỗi `s1` và `s2` theo mã ASCII, có phân biệt chữ hoa chữ thường. Hàm trả về một giá trị `int`:
  - $< 0$  nếu  $s1 < s2$
  - $== 0$  nếu  $s1 == s2$
  - $> 0$  nếu  $s1 > s2$

So sánh chuỗi không phân biệt hoa thường dùng `stricmp`

## 5. Các hàm chuẩn xử lý xâu ký tự (tiếp)

- ✧ Hàm đảo xâu: `strrev(s)` đảo ngược các ký tự trong xâu `s`, đầu về cuối, cuối về đầu.
- ✧ Hàm chuyển chữ thường thành chữ hoa: `strupr(s)` chuyển các chữ cái thường trong xâu `s` thành chữ hoa, các chữ khác không thay đổi.
- ✧ Hàm chuyển chữ hoa thành chữ thường: `strlwr(s)` chuyển các chữ cái hoa trong xâu `s` thành chữ thường, các chữ khác không thay đổi.

## 6. Mảng chuỗi ký tự

- ✧ Một mảng chuỗi ký tự rất hay được sử dụng, chẳng hạn như dùng để lưu trữ danh sách tên, danh sách mật khẩu, danh sách tên tệp,...
- ✧ Để tạo mảng các biến chuỗi rỗng ta tạo một mảng hai chiều bởi vì chuỗi ký tự cũng là một mảng và mảng chuỗi ký tự thực chất là mảng của các mảng.
- ✧ Ví dụ: để lưu trữ 5 họ tên, mỗi họ tên có tối đa 20 ký tự ta định nghĩa mảng chuỗi như sau:  
`char names[5][21];` Đoạn chương trình dưới đây cho phép người sử dụng nhập vào các họ tên để lưu trong mảng trên.

## 6. Mảng chuỗi ký tự (tiếp)

```
for(int i=0;i<5;++i)
{
    cout<<"Nhập vào một họ tên (ấn enter để thoát: ";
    cin.get(names[i],sizeof(names[i]));
}
```



## 6. Mảng xâu ký tự (tiếp)

✧ Ta cũng có thể khởi tạo mảng xâu ngay khi định nghĩa giống như các mảng khác. Ví dụ:


```
char Thu[7][ ] =
```


```
{"Thu Hai", "Thu Ba", "Thu Tu", "Thu Nam",  
"Thu Sau", "Thu Bay", "Chu Nhat"};
```

# Ví dụ

- 1) Nhập vào một số nguyên dương, đưa ra chuỗi ký tự số hex tương ứng.
- 2) Đọc vào từ tệp văn bản một danh sách n tên (không có họ đệm). Sắp xếp danh sách tên theo vần ABC. Ghi danh sách đã sắp xếp ra tệp văn bản.

# Bài tập chương 6

 ✧ Bài 1. Viết chương trình nhập vào một dãy  $n$  số nguyên, hãy sắp xếp dãy số này theo thứ tự không giảm bằng phương pháp sắp xếp chọn.

 ✧ Bài 2. Hình vuông kỳ ảo bậc  $n$  được định nghĩa là một ma trận vuông cấp  $n$  sao cho:

- Chứa đủ  $n^2$  số tự nhiên đầu tiên  $(1, 2, 3, \dots, n^2)$
- Tổng các số trên từng hàng bằng tổng các số trên từng cột bằng tổng các số trên đường chéo chính bằng tổng các số trên đường chéo phụ.

Viết chương trình nhập vào số tự nhiên lẻ  $n$ , đưa ra màn hình một hình vuông kỳ ảo bậc  $n$  lẻ đó.



# Bài tập chương 6 (tiếp)

Ví dụ dưới đây là 2 hình vuông kỳ ảo bậc 3 và bậc 5:

8	1	6
3	5	7
4	9	2

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

# Bài tập chương (tiếp)

-  ✧ Bài 3. Viết chương trình nhập vào một số nguyên dương  $n$ , đưa ra màn hình xâu ký tự số nhị phân của  $n$ .
-  ✧ Bài 4. Hai từ  $x$  và  $y$  gọi là anagram với nhau nếu mỗi ký tự của từ này cũng có mặt trong từ kia (không phân biệt chữ hoa chữ thường) và hơn nữa số lượng từng loại ký tự xuất hiện trong hai từ là bằng nhau. Ví dụ các từ sau là anagram của nhau: read, dear, dare. Viết chương trình nhập vào 2 từ  $x$  và  $y$  rồi kiểm tra xem chúng có phải là anagram của nhau không.

# Bài tập chương (tiếp)



Bài 5. Viết chương trình nhập vào một danh sách  $n$  tên. Sắp xếp tên theo vần ABC. Đưa danh sách tên ra màn hình theo dạng cột.