

## Mảng

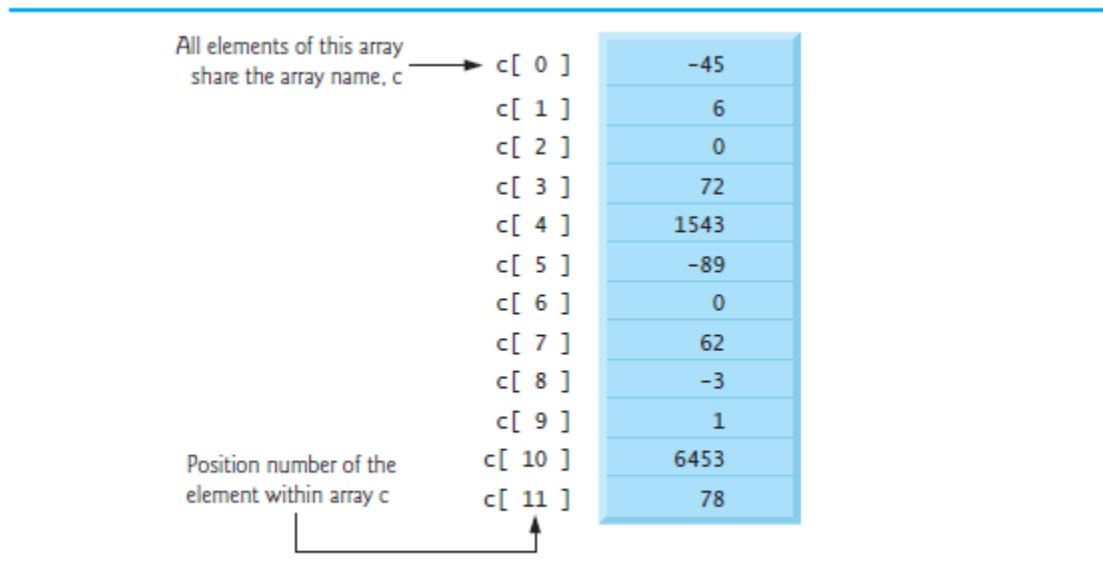
### Nội dung chính:

- Khái niệm liên quan tới mảng.
- Định nghĩa, khởi tạo, gán giá trị, duyệt các phần tử mảng.
- Truyền mảng vào hàm và trả về mảng từ một hàm.
- Sắp xếp các phần tử của mảng.
- Tìm kiếm trong mảng.
- Mảng hai chiều.

### 1. Khái niệm:

- Mảng là một nhóm các ô nhớ liên kề nhau dùng để lưu trữ các phần tử cùng kiểu. Vậy mảng có kiểu là kiểu của các phần tử mà nó chứa. Mảng là một kiểu dữ liệu có cấu trúc.
- Mỗi thành phần cấu thành nên mảng gọi là một phần tử của mảng.
- Để truy cập một phần tử cụ thể của mảng, ta cần xác định tên mảng và vị trí của phần tử đó trong mảng, vị trí này gọi là chỉ số của mảng.

Ví dụ:



**Fig. 6.1** | 12-element array.

- Chỉ số mảng bắt đầu từ 0, như vậy mảng có n phần tử thì phần tử thứ n sẽ có chỉ số là n-1.

- Giả dụ ta muốn truy cập phần tử đầu tiên của mảng a thì cú pháp a[0] là đúng, trong khi cú pháp a[1] là không chính xác dù có thể hợp lệ. [Tại sao?](#)

## 2. Định nghĩa, khởi tạo, gán giá trị và duyệt các phần tử mảng.

- Cú pháp tổng quát để định nghĩa (khai báo) mảng:

*Kiểu\_tên\_mảng[số\_lượng\_phần\_tử];*

Trong đó:

- Kiểu là kiểu dữ liệu hợp lệ trong ngôn ngữ C,
- Tên\_mảng do người dùng tự đặt, tránh keyword và đặt tên theo quy tắc tên biến
- Số\_lượng\_phần\_tử\_mảng: là số nguyên dương.

Ví dụ:

- int a[10];
- char c[20];
- double arr[12];
- Khởi tạo: có thể khởi tạo các phần tử ban đầu cho mảng: ta có thể khởi tạo giá trị ban đầu ngay khi định nghĩa mảng hoặc gán giá trị về sau khi cần thiết. Cú pháp khởi tạo tổng quát là:
  - *kiểu\_tên\_mảng[]={danh\_sách\_giá\_trị}.*
  - Trong đó danh sách giá trị có từ 2 giá trị trở lên sẽ phân tách nhau bằng dấu phẩy(.). Có thể xác định sẵn số phần tử cho mảng ngay từ đầu nếu khởi tạo. Nếu danh sách giá trị không đủ, giá trị mặc định tương ứng sẽ được điền vào tương ứng từng kiểu dữ liệu: các kiểu số là 0, kiểu char là kí tự null('\0').

Ví dụ:

- int a[] = {1, 20, 5, 6, -7, -11};
- char greeting[] = {'H', 'e', 'l', 'l', 'o'};
- int arr[10] = {1, 2, 6 };// các phần tử còn lại sẽ là 0
- Trường hợp không khởi tạo thì bắt buộc phải xác định rõ số lượng phần tử của mảng. Ví dụ: int arrOfInt[100];
- Có thể gán giá trị riêng lẻ cho từng phần tử của mảng. Ví dụ:
  - int arrOfInt[100]; sau đó gán arrOfInt[2] = 200; arrOfInt[10] = -230; ...
- Để duyệt mảng thường sử dụng vòng lặp for.

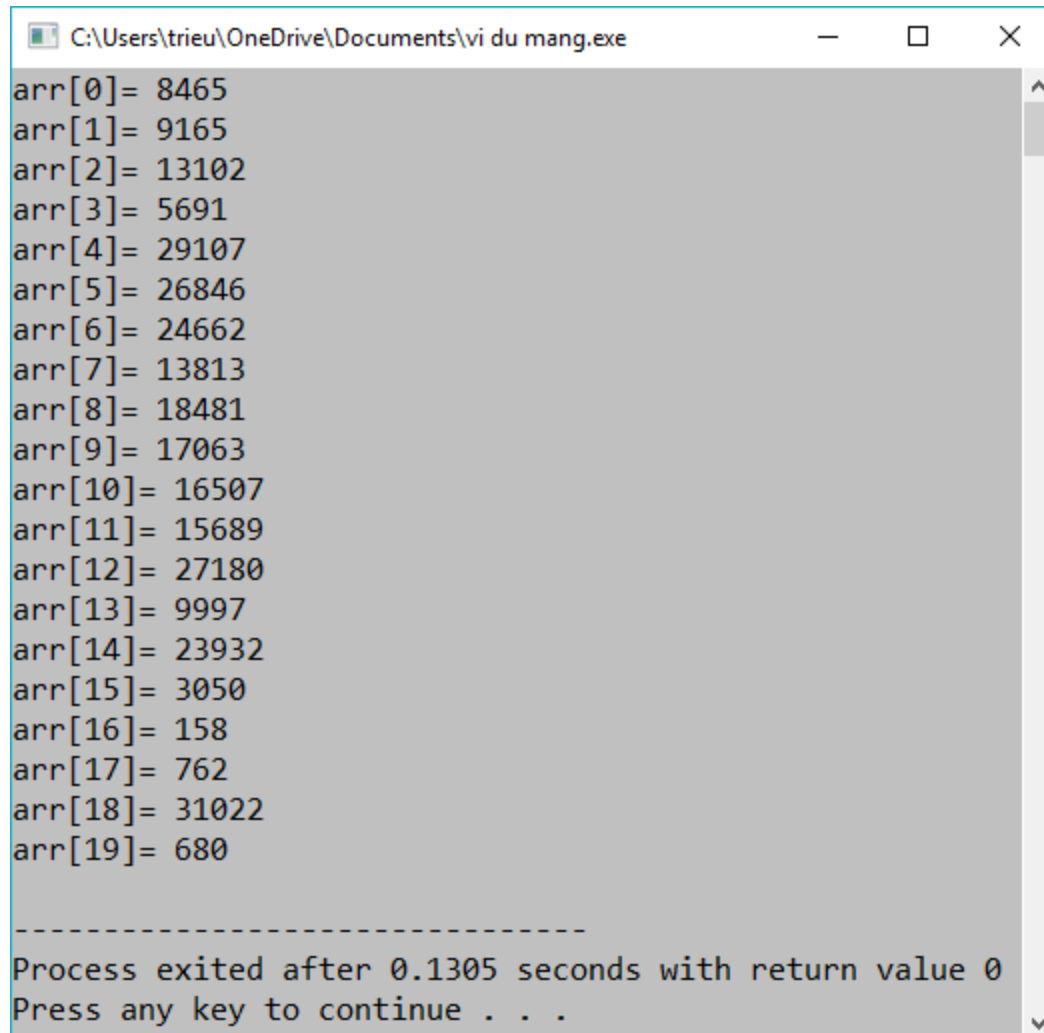
Lưu ý: trong quá trình gán và duyệt các phần tử mảng, chỉ số mảng không được vượt quá n-1, với n là số lượng phần tử mảng đã xác định từ trước. trong ví dụ [int arrOfInt\[100\];](#) thì giá trị chỉ số tối đa là 99 tức là 100 – 1.



Ví dụ: chương trình tạo mảng arr kiểu int có 20 phần tử, sau đó gán số ngẫu nhiên cho từng phần tử rồi thực hiện hiển thị kết quả sau khi gán:

```
2 #include<stdlib.h>
3 #include<time.h>
4
5 #define MAX 20
6
7 int main()
8 {
9     srand(time(NULL));
10    int i;
11    int arr[MAX]; // định nghĩa mảng kiểu int có MAX phần tử
12
13    // thực hiện gán giá trị cho các phần tử mảng arr[]
14    for( i = 0; i < MAX; i++ ){
15        arr[i] = rand(); // gán cho phần tử thu i một giá trị ngẫu nhiên
16    }
17
18    // hiển thị thông tin các phần tử trong mảng
19    for( i = 0; i < MAX; i++ ){
20        printf("arr[%d]= %d\n", i, arr[i]);
21    }
22
23    return 0;
24 }
```

Kết quả thực hiện:



```
C:\Users\trieu\OneDrive\Documents\vi du mang.exe
arr[0]= 8465
arr[1]= 9165
arr[2]= 13102
arr[3]= 5691
arr[4]= 29107
arr[5]= 26846
arr[6]= 24662
arr[7]= 13813
arr[8]= 18481
arr[9]= 17063
arr[10]= 16507
arr[11]= 15689
arr[12]= 27180
arr[13]= 9997
arr[14]= 23932
arr[15]= 3050
arr[16]= 158
arr[17]= 762
arr[18]= 31022
arr[19]= 680

-----
Process exited after 0.1305 seconds with return value 0
Press any key to continue . . .
```

3. Truyền mảng vào hàm và trả về mảng từ một hàm.
4. Sắp xếp các phần tử mảng.
5. Tìm kiếm trong mảng.
6. Mảng 2 chiều