```c
#include <stdio.h>

int main()
{
int arr[5];
//assume base address of arr is 2000 and size of integer is 32 bit
printf("%u %u",arr+1,&arr+1);
return 0;
}
```

Giả định

- Địa chỉ cơ sở (địa chỉ đầu tiên) của mảng `arr` là `2000`.
- Mỗi phần tử `int` có kích thước 4 byte (do một `int` là 32 bit), vì vậy mỗi bước dịch con trỏ sẽ tăng thêm 4 byte khi làm việc với con trỏ `int`.

Giải thích chi tiết

1. `arr` và `arr + 1`
   - `arr` là tên của mảng và khi sử dụng như một con trỏ, nó trỏ đến phần tử đầu tiên trong mảng (tức là `&arr[0]`).
   - `arr + 1` là một phép toán con trỏ. Khi tăng `arr` lên `1`, địa chỉ sẽ được tăng lên 4 byte (kích thước của một `int`).
   - Vậy `arr + 1` sẽ trỏ đến địa chỉ `2000 + 4 = 2004`, tức là phần tử thứ hai của mảng `arr`.

2. `&arr` và `&arr + 1`
   - `&arr` là địa chỉ của toàn bộ mảng `arr`, không phải chỉ một phần tử của mảng. Kích thước của `arr` là 5 phần tử `int`, nên tổng kích thước của nó là `5 * 4 = 20` byte.
   - `&arr + 1` sẽ dịch chuyển địa chỉ `&arr` lên một "khoảng" bằng toàn bộ mảng `arr`, tức là tăng lên `20` byte.
   - Vì `&arr` có địa chỉ ban đầu là `2000`, nên `&arr + 1` sẽ trỏ đến địa chỉ `2000 + 20 = 2020`.
3. `printf("%u %u", arr + 1, &arr + 1);`
   - `arr + 1` sẽ in ra `2004`, vì nó trỏ đến phần tử thứ hai của mảng `arr`.
   - `&arr + 1` sẽ in ra `2020`, vì nó trỏ đến địa chỉ ngay sau toàn bộ mảng `arr`.

Kết quả đầu ra

Kết quả khi chạy chương trình sẽ là:

yaml Sao chép mã

```
2004 2020
```

Tóm lại

- `arr + 1` dịch chuyển con trỏ `arr` lên `4` byte, trỏ đến phần tử thứ hai của mảng.
- `&arr + 1` dịch chuyển địa chỉ của toàn bộ mảng `arr` lên `20` byte, đến vị trí ngay sau mảng.