

Con trỏ và mảng một chiều

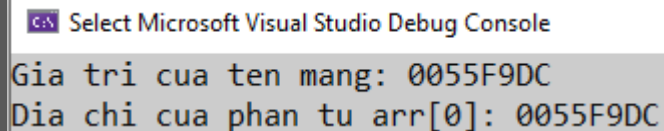
- Bản chất của mảng 1 chiều
- Các phép toán với con trỏ
- Sử dụng toán tử sizeof
- Ví dụ minh họa



Bản chất mảng 1 chiều

- Con trỏ và mảng 1 chiều có mối quan hệ mật thiết và có thể sử dụng thay thế cho nhau
- Bản chất của mảng 1 chiều là một con trỏ luôn trỏ đến địa chỉ cố định, đó là địa chỉ của phần tử đầu tiên trong mảng
- Hãy xem ví dụ sau, tất cả cho cùng 1 kết quả: địa chỉ của phần tử đầu tiên trong mảng!

```
int arr[] = { 1, 2, 3, 4, 5 };  
  
printf("Gia tri cua ten mang: %p\n", arr);  
printf("Dia chi cua phan tu arr[0]: %p\n", &arr[0]);
```



```
Select Microsoft Visual Studio Debug Console  
Gia tri cua ten mang: 0055F9DC  
Dia chi cua phan tu arr[0]: 0055F9DC
```

Dùng con trỏ thay mảng

- Ta có thể gán trực tiếp tên mảng cho con trỏ cùng kiểu và dùng con trỏ thay mảng:

```
int arr[] = { 1, 2, 3, 4, 5 };  
int* aPtr = arr; // gán tên mảng cho con trỏ  
// hiển thị các phần tử trong mảng:  
puts("Các phần tử trong mảng: ");  
for (size_t i = 0; i < 5; i++) {  
    printf("%5d", aPtr[i]);  
}
```

- Khi truyền mảng vào hàm thì ta có thể sử dụng con trỏ để làm tham số thay cho mảng

Các phép toán với con trỏ

- Khi dùng con trỏ thay mảng ta có các phép toán sau: =, ++, --, +, +=, -, -= và các phép so sánh. Cụ thể:
- Phép gán(=): ta có thể gán địa chỉ của một biến hoặc một con trỏ cho 1 con trỏ cùng kiểu. Ví dụ `aPtr = arr`
- Phép +, +=, -, -=, ++, --: khi áp dụng với con trỏ sẽ làm con trỏ nhảy đến địa chỉ cách địa chỉ hiện tại một lượng là x tương ứng.

```
int arr[] = { 1, 2, 3, 4, 5 };
int* aPtr = arr; // gán con trỏ arr cho aPtr
// hiển thị các phần tử trong mảng:
puts("Các phần tử trong mảng: ");
for (size_t i = 0; i < 5; i++) {
    printf("%5d", *(aPtr++));
}
```

Các phép toán với con trỏ

- Ví dụ khác:

```
int arr[] = { 1, 2, 3, 4, 5 };  
int* aPtr = arr; // gán con trỏ arr cho aPtr  
// hiển thị các phần tử trong mảng:  
puts("Các phần tử trong mảng: ");  
for (size_t i = 0; i < 5; i++) {  
    printf("%5d", *(aPtr + i));  
}
```

- Khi thực hiện $aPtr + i$ thì $aPtr$ sẽ trỏ đến vùng nhớ cách vùng nhớ hiện tại 1 lượng là $i \times (\text{kích thước kiểu của } aPtr)$. Hay nói cách khác là trỏ đến vị trí phần tử thứ i của mảng

Các phép toán với con trỏ

- Khi thao tác con trỏ trỏ đến mảng 1 chiều ta còn có thể so sánh địa chỉ của các phần tử, ví dụ:

```
int arr[] = { 1, 2, 3, 4, 5 };  
int* aPtr = arr; // gán con trỏ arr cho aPtr  
// hiển thị các phần tử trong mảng:  
puts("Các phần tử trong mảng: ");  
for (; aPtr <= &arr[4]; aPtr++) {  
    printf("%5d", *aPtr);  
}
```

- Nhấn mạnh rằng chỉ áp dụng phép so sánh địa chỉ con trỏ khi áp dụng con trỏ với mảng 1 chiều

Lưu ý quan trọng

- Không thực hiện các phép toán vừa đề cập với các con trỏ trỏ đến địa chỉ của biến bình thường
- Không cố gắng thay đổi tên mảng để trỏ đến phần tử khác, ví dụ `ar += 3; // error!`
- Có thể dùng con trỏ void để gán cho bất kì kiểu con trỏ nào và ngược lại
- Con trỏ void không có kích thước cụ thể nên không thể sử dụng toán tử phân giải địa chỉ với nó.
- Ví dụ: `void* vPtr = arr;`
`*(vPtr) = 50; // error!`

Toán tử sizeof

- sizeof là toán tử trả về kích thước của biến ở số byte
- Nếu áp dụng sizeof với biến thường, hoặc mảng sẽ cho tổng kích thước của biến/mảng
- Nhưng nếu áp dụng với con trỏ, ta sẽ nhận cùng giá trị: 4byte với hệ điều hành windows, linux và 8byte với Mac.

- Ví dụ:

```
int* aPtr;  
long long* llPtr;  
char* cPtr;  
printf("Kích thước con trỏ char: %u\n", sizeof(cPtr));  
printf("Kích thước con trỏ long long: %u\n", sizeof(llPtr));  
printf("Kích thước con trỏ int: %u\n", sizeof(aPtr));
```

Kết quả:

```
Kích thước con trỏ char: 4  
Kích thước con trỏ long long: 4  
Kích thước con trỏ int: 4
```




Tiếp theo

Sử dụng keyword const