

Con trỏ

huydq@soict.hust.edu.vn

Địa chỉ bộ nhớ của biến

```
char ch = 'A' ;
```

ch :

0x2000

'A'

Địa chỉ bộ nhớ
của biến *ch*

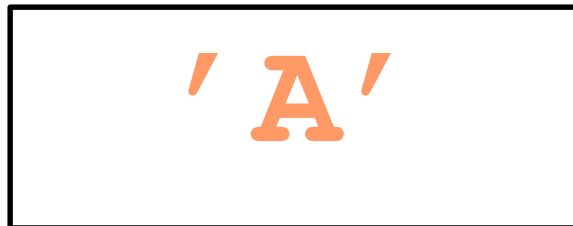
Giá trị của biến
ch

Toán tử &

- Trả về địa chỉ bộ nhớ của một đối tượng

```
char ch = 'A';
```

0x2000

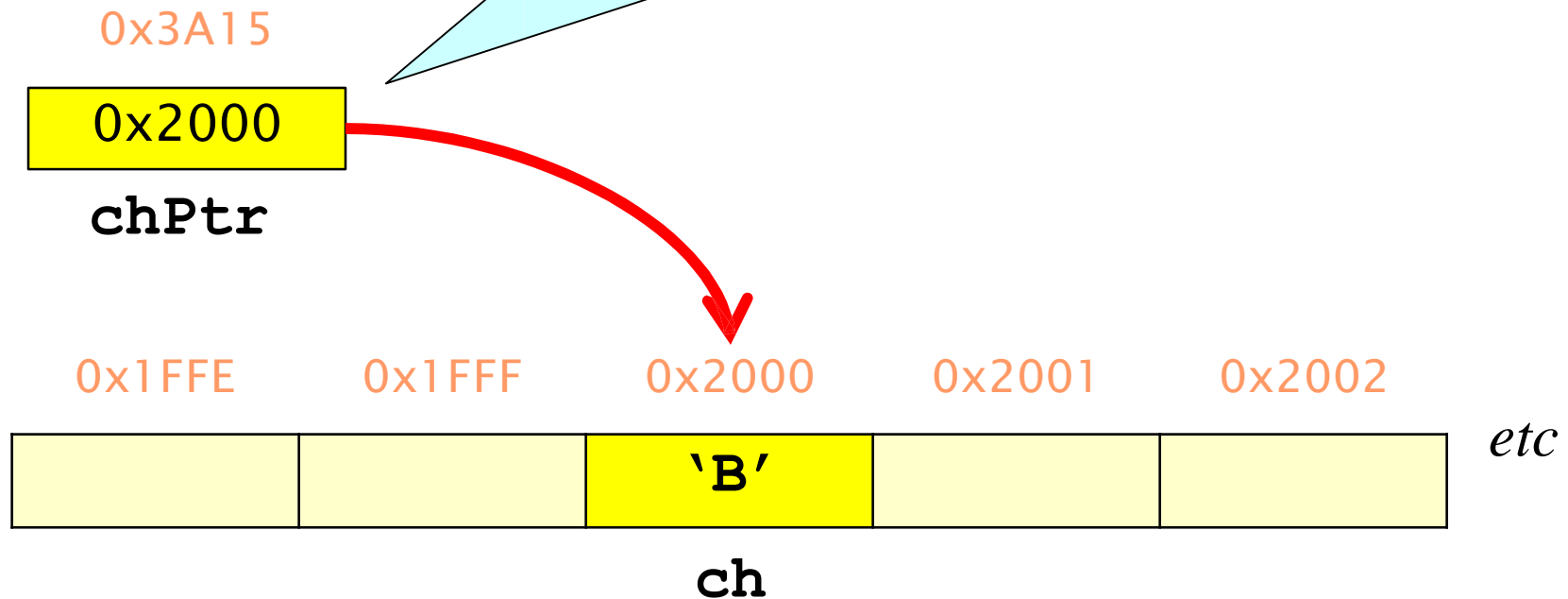


&ch

trả về giá trị 0x2000

Con trỏ

Một biến có thể lưu giá trị là **địa chỉ bộ nhớ** của một biến khác



Khai báo con trỏ

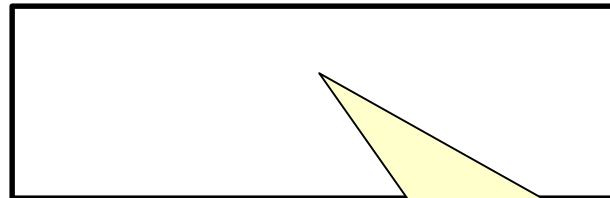
- Khai báo con trỏ là **một biến**
- Chứa một **địa chỉ bộ nhớ**
- Thường trỏ tới **một kiểu dữ liệu** xác định

Ví dụ:

```
char* cPtr;
```

cPtr:

0x2004



Có thể chứa địa chỉ của
biến kiểu **char**

Khai báo con trỏ (tiếp)

- Có thể tạo con trỏ đến biến có kiểu dữ liệu bất kì

Ví dụ:

```
int * numPtr;  
float * xPtr;
```

- Một biến con trỏ luôn được khai báo đi kèm với toán tử *

Ví dụ:

```
int *numPtr1, *numPtr2;  
float *xPtr, *yPtr;
```

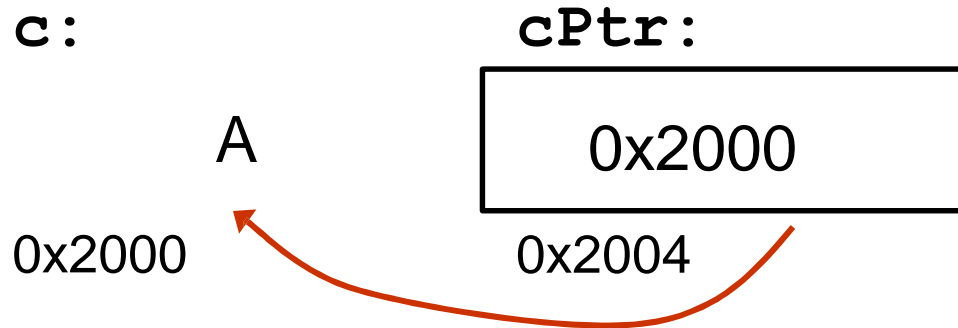
Tham chiếu

- Dùng toán tử & để xác lập địa chỉ tham chiếu cho con trỏ

Ví dụ:

```
char c = 'A';  
char *cPtr;  
cPtr = &c;
```

*Xác lập địa chỉ của **c** cho
con trỏ **cPtr***



Chú ý về con trỏ

- Chỉ có thể xác lập tham chiếu cho con trỏ tới địa chỉ của biến có kiểu tương thích với con trỏ

Ví dụ:

```
int   aNumber;  
char *ptr;
```

Lỗi tương thích
về kiểu dữ liệu

```
ptr = &aNumber;
```

- Để in giá trị địa chỉ lưu bởi một con trỏ ta có thể sử dụng định dạng in **%p**

Ví dụ:

```
printf("%p", ptr);
```

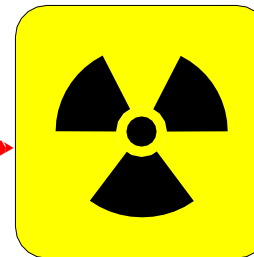

Con trỏ NULL

```
int *numPtr;
```

Chú ý con trỏ không được khởi tạo giá trị tham chiếu!

???

numPtr



- Khởi tạo con trỏ với giá trị **NULL** để chắc chắn không sử dụng tham chiếu sai trong chương trình

```
int *numPtr = NULL;
```

NULL

numPtr

Con trỏ có tham chiếu NULL (không chứa địa chỉ nào cả)

Toán tử *

- Cho phép truy cập biến có địa chỉ bộ nhớ lưu bởi một con trỏ
- Được biết đến như toán tử dùng để “**khử tham chiếu**” cho con trỏ
- Tránh nhầm lẫn với toán tử * dùng trong khai báo con trỏ

Ví dụ:

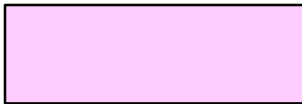
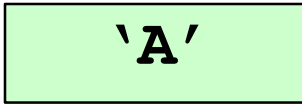
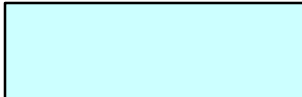
```
char c = 'A';  
char *cPtr = NULL;  
cPtr = &c;  
*cPtr = 'B';
```

đổi giá trị của biến **c** trở
bởi **cPtr**

Các bước sử dụng con trỏ

- **Bước 1:** Khai báo biến được trỏ bởi con trỏ

```
int  num; char  
ch = 'A';  
float x;
```

num:	
ch:	
x:	

Các bước sử dụng con trỏ (tiếp)

- **Bước 2:** Khai báo biến con trỏ

```
int num;  
char ch = 'A';  
float x;
```

```
int* numPtr = NULL;  
char *chPtr = NULL;  
float *xPtr = NULL;
```

numPtr:

NULL

chPtr:

NULL

xPtr:

NULL

num:

--

ch:

'A'

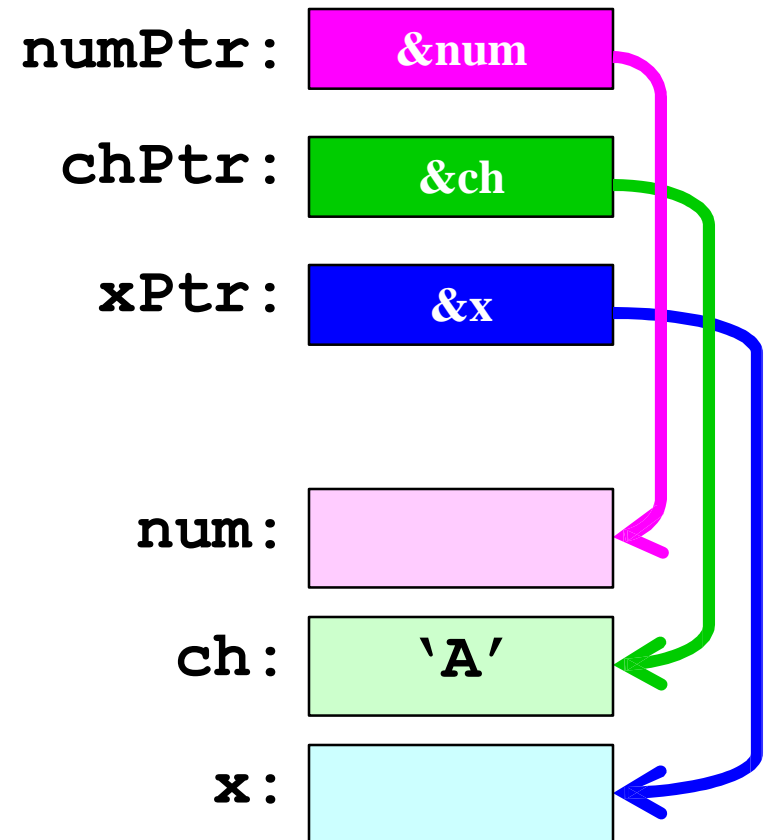
x:

--

Các bước sử dụng con trỏ (tiếp)

- **Bước 3:** Xác lập tham chiếu cho con trỏ

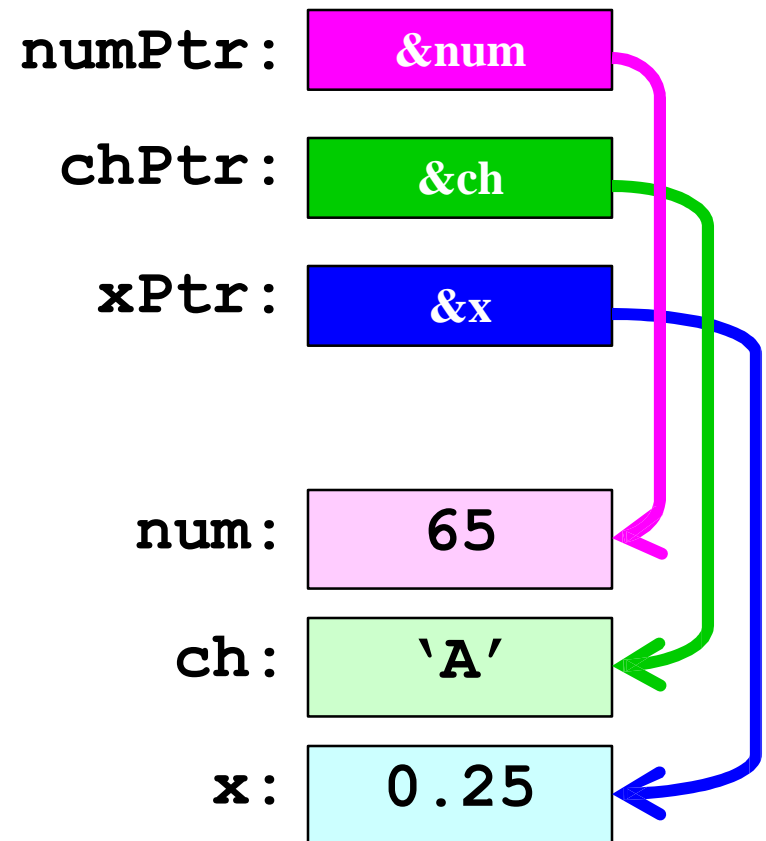
```
int    num;  
char   ch = 'A';  
float  x;  
  
int*    numPtr = NULL;  
char*   chPtr  = NULL;  
float*  xPtr   = NULL;  
  
numPtr = &num;  
chPtr  = &ch;  
xPtr   = &x;
```



Các bước sử dụng con trỏ (tiếp)

- **Step 4:** Khử tham chiếu con trỏ

```
int    num;  
char   ch = 'A';  
float  x;  
  
int*    numPtr = NULL;  
char*   chPtr  = NULL;  
float*  xPtr   = NULL;  
  
numPtr = &num;  
chPtr  = &ch;  
xPtr   = &x;  
  
*xPtr = 0.25;  
*numPtr = *chPtr;
```

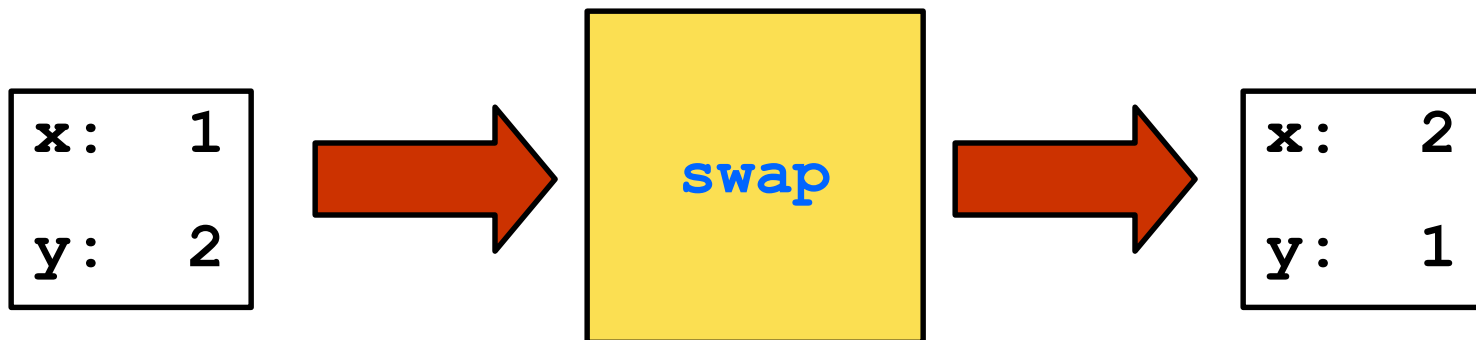


Lỗi thường gặp

- Không thể tham chiếu một con trỏ đến một hằng số hay một biểu thức
- Cũng không thể thay đổi địa chỉ của một biến trong bộ nhớ (bởi vì nó không được quyết định bởi người sử dụng!)
- Do vậy sau đây là một số lỗi:
 - `ptr = &3;`
 - `ptr = &(x+5);`
 - `&x = ptr;`
 - `&x = 0x2000;`

Tham số hàm và con trỏ

- Ví dụ : Tạo hàm là thay đổi giá trị của hai biến truyền vào



Truyen theo tham t..

```
#include <stdio.h>
```

```
void swap1(int a, int b)
```

```
{
```

```
    int tmp;
```

```
    tmp = a;
```

```
    a = b;
```

```
    b = tmp;
```

```
    return;
```

```
}
```

```
int main()
```

```
{
```

```
    int x = 1, y = 2;
```

```
    swap1(x, y);
```

```
    printf("%d %d\n", x, y);
```

```
    return 0;
```

```
}
```

tmp:

a:

b:

x:

y:

Truyen theo tham t..(tiep)

```
#include <stdio.h>
```

```
void swap1(int a, int b)
{
    int tmp;

    tmp = a;
    a = b;
    b = tmp;
    return;
}
```

```
int main()
{
    int x = 1, y = 2;

    swap1(x, y);
    printf("%d %d\n", x, y);
    return 0;
}
```

tmp:

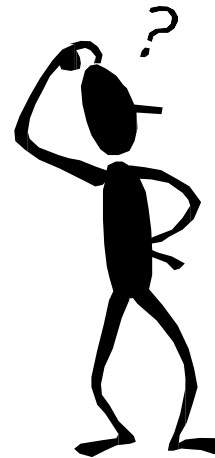
1

a:

2

b:

1



x:

1

y:

2

Truyen theo tham chieu

```
#include <stdio.h>
```

```
void swap2(int* a, int* b) tmp:
```

```
{
```

```
    int tmp;
```

```
    tmp = *a;
```

```
    *a = *b;
```

```
    *b = tmp;
```

```
    return;
```

```
}
```

```
int main()
```

```
{
```

```
    int x = 1, y = 2;
```

```
    swap2(&x, &y);
```

```
    printf("%d %d\n", x, y);
```

```
    return 0;
```

```
}
```

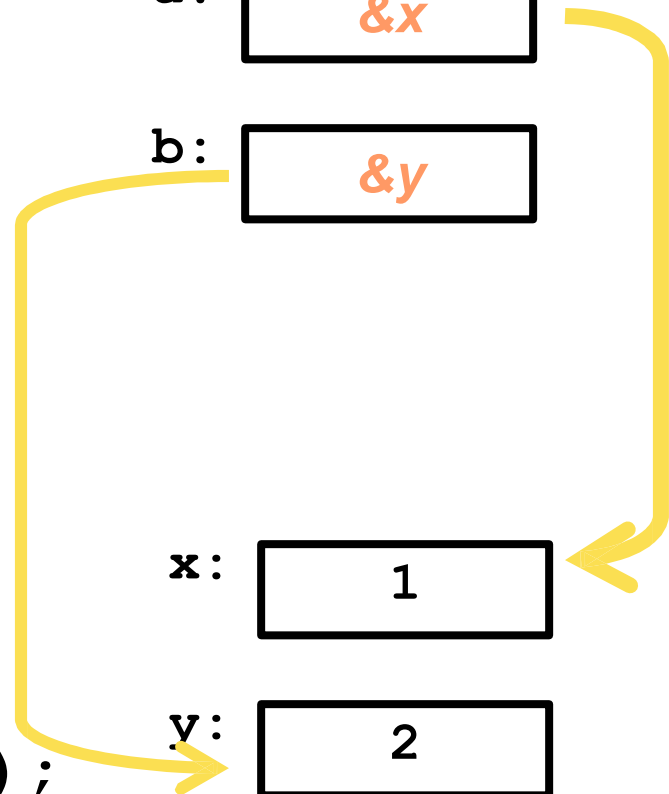
tmp: 

a: 

b: 

x: 

y: 



Truyen theo tham chieu (tiep)

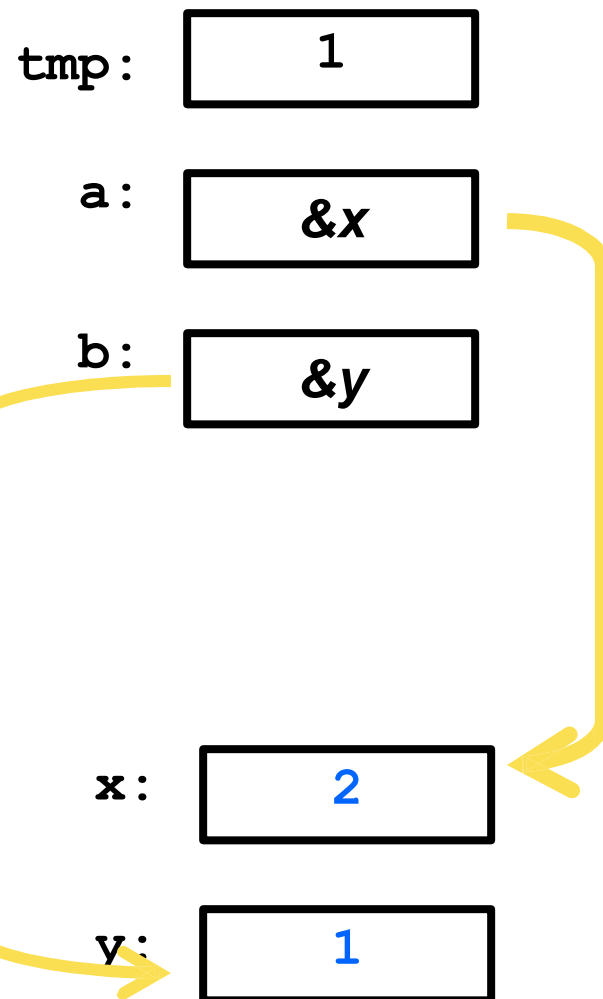
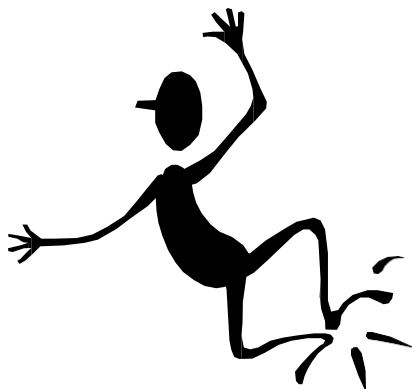
```
#include <stdio.h>
```

```
void swap2(int* a, int* b)
{
    int tmp;

    tmp = *a;
    *a = *b;
    *b = tmp;
    return;
}
```

```
int main()
{
    int x = 1, y = 2;

    swap2(&x, &y);
    printf("%d %d\n", x, y);
    return 0;
}
```



Tham số hàm là con trỏ

- Cho phép thay đổi giá trị của một biến qua lời gọi hàm
- Truyền tham chiếu trong hàm **scanf**

```
char ch;  
int numx;  
float numy;  
scanf("%c %d %f", &ch, &numx, &numy);
```



Ưu điểm

- Truyền theo tham chiếu sẽ hiệu quả hơn truyền theo tham trị bởi chúng ta không mất thời gian tạo bản sao giá trị truyền vào cho hàm mỗi lần gọi
- Có thể sử dụng tham số dạng tham chiếu để tạo một hàm trả về nhiều hơn một giá trị

```
void maxmin(int a, int b,  
            int *max, int *min)  
{  
    *max = (a>b) ? a : b;  
    *min = (a<b) ? a : b;  
}
```

Nhược điểm

- Rất khó kiểm soát một chương trình sử dụng nhiều hàm với tham số con trỏ vì khi đó một biến có thể bị thay đổi ở bất kì đâu trong chương trình.
- Chỉ dùng hàm với tham số dạng tham chiếu (con trỏ) khi cần thiết thay đổi giá trị một biến được truyền vào cho hàm.

Mảng và con trỏ

- Chú ý rằng một mảng tương ứng với địa chỉ phần tử đầu tiên của nó
- Do vậy một mảng A là con trỏ đến phần tử A[0]

Ví dụ:

```
int A[10];  
int *ptr;  
ptr = A; /* ptr = &A[0] */
```

- Có sự tương đương giữa ptr và A vì cùng mang một địa giống nhau
- Không thể thay đổi địa chỉ một mảng nhưng lại có thể thay đổi địa chỉ của con trỏ

Ví dụ:

```
int B[10];  
ptr = B; /* OK */  
A = B;   /* KO */
```


Mảng và con trỏ (tiếp)

- Có thể truy cập vào các phần tử của một mảng thông qua con trỏ

Ví dụ:

```
int A[10];  
int *ptr=A;  
ptr[2] = 5; /* A[2] = 5 */
```

- Con trỏ có thể được tăng hoặc giảm trỏ đến phần tử khác trong mảng
- Nếu **p** là một con trỏ tới một kiểu xác định, **p+1** đưa ra địa chỉ chính xác của biến tiếp theo trong bộ nhớ có cùng kiểu
- **p++**, **p--**, hay **p += i** cũng đem lại ý nghĩa tương tự

Ví dụ:

```
ptr += 2; /* ptr → A[2] */  
ptr[1] = 3; /* A[3] = 5 */  
A[0] = *(ptr+1); /* A[0] = A[3] */
```

Truyền mảng cho hàm

- Một mảng truyền cho hàm tương ứng với địa chỉ gốc của nó
- Do đó có hai cách tương đương để khai báo hàm có thể truyền tham số là mảng

– `f (int array[])`

Như một mảng có kích thước không xác định

hoặc

– `f (int *ptr)`

Như một con trỏ

Chương trình dãy số (v2)

```
#include <stdio.h>

void nhapMang(int *ptr, int num)
{
    int i;
    for(i=0; i<num; i++) {
        printf("Phan tu thu %d:", i+1);
        scanf("%d", ptr+i);
    }
}
```

Lấy địa chỉ phần tử
chỉ số i qua con trỏ

```
void inNguoc(int *ptr, int num)
{
    int i;
    for(i=num-1; i>=0; i--)
        printf("%5d", ptr[i]);
}
```

Lấy giá trị phần tử
chỉ số i qua con trỏ

Chương trình dãy số (v2)

```
int main(void)
{
    int n, A[10];
    printf("Nhap so phan tu trong day (n<=10):");
    scanf("%d",&n);

    printf("Nhap cac phan tu trong day:¥n");
    nhapMang(A, n);

    printf("Day so sau khi dao lai:¥n");
    inNguoc(A, n);

    return 0;
}
```

Chương trình chính
vẫn không thay đổi khi
thay cách khai báo
hàm