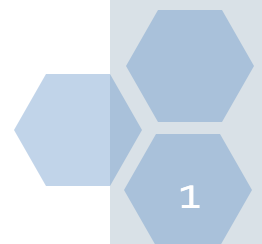
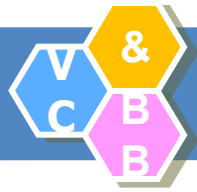


Nhập môn Lập trình Con trỏ (cơ bản)





Nội dung

1

Khái niệm và cách sử dụng

2

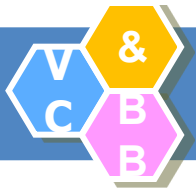
Các cách truyền đối số cho hàm

3

Con trỏ và mảng một chiều

4

Con trỏ và cấu trúc



Kiến trúc máy tính

❖ Bộ nhớ máy tính

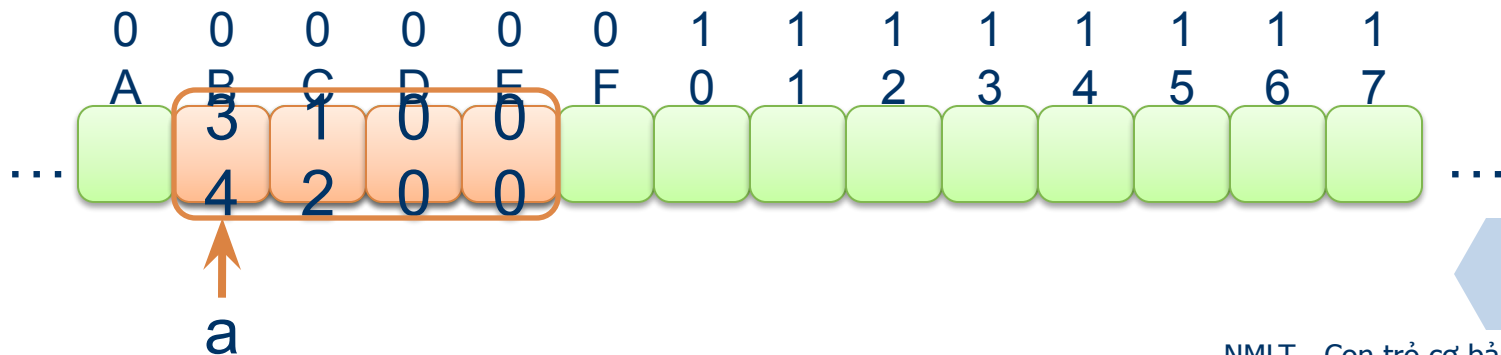
- Bộ nhớ RAM chứa rất **nhiều ô nhớ**, mỗi ô nhớ có **kích thước 1 byte**.
- RAM dùng để chứa **một phần hệ điều hành, các lệnh chương trình, các dữ liệu...**
- Mỗi ô nhớ có **địa chỉ duy nhất** và địa chỉ này được **đánh số từ 0** trở đi.
- Ví dụ
 - RAM **512MB** được đánh địa chỉ từ **0** đến **$2^{29} - 1$**
 - RAM **2GB** được đánh địa chỉ từ **0** đến **$2^{31} - 1$**

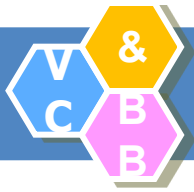


Khai báo biến trong C

- ❖ Quy trình xử lý của trình biên dịch
 - Dành riêng một vùng nhớ với địa chỉ duy nhất để lưu biến đó.
 - Liên kết địa chỉ ô nhớ đó với tên biến.
 - Khi gọi tên biến, nó sẽ truy xuất tự động đến ô nhớ đã liên kết với tên biến.

❖ Ví dụ: `int a = 0x1234;` // Giả sử địa chỉ `0x0B`

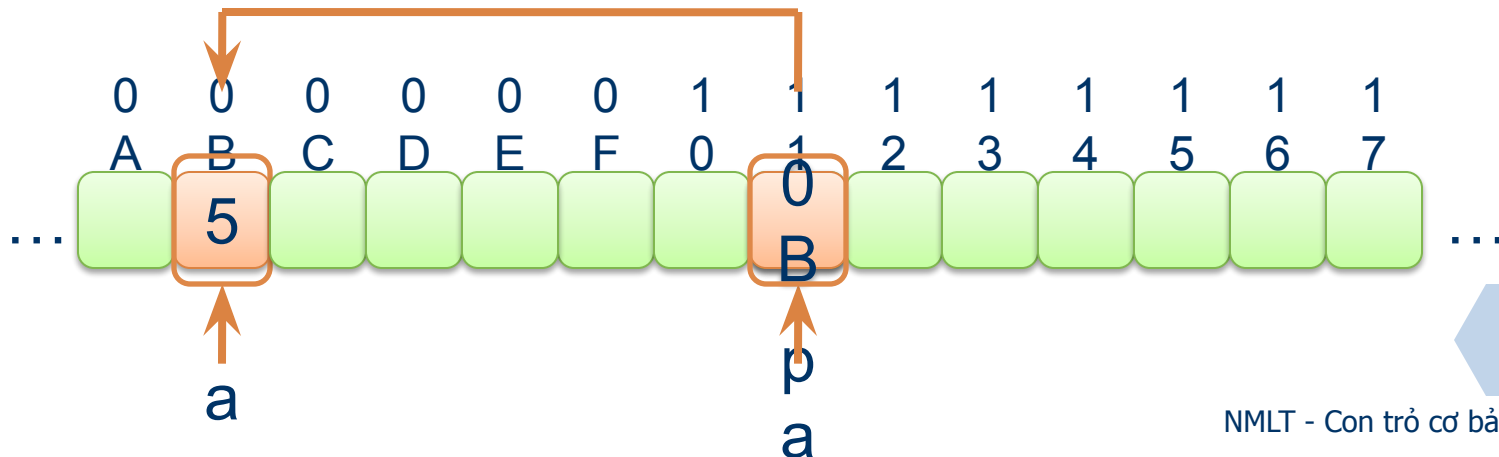


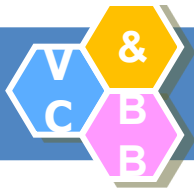


Khái niệm con trỏ

❖ Khái niệm

- Địa chỉ của biến là một con số.
- Ta có thể tạo **biến khác để lưu địa chỉ của biến này** □ Con trỏ.





Khai báo con trỏ

❖ Khai báo

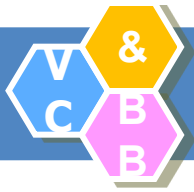
- Giống như mọi biến khác, biến con trỏ muốn sử dụng cũng cần phải được khai báo

```
<kiểu dữ liệu> * <tên biến con trỏ>;
```

❖ Ví dụ

```
char *ch1, *ch2;  
int *p1, p2;
```

- ch1 và ch2 là biến con trỏ, trỏ tới vùng nhớ kiểu char (1 byte).
- p1 là biến con trỏ, trỏ tới vùng nhớ kiểu int (4 bytes) còn p2 là biến kiểu int bình thường.



Khai báo con trỏ

❖ Sử dụng từ khóa typedef

```
typedef <kiểu dữ liệu> *<tên kiểu con trỏ>;
```

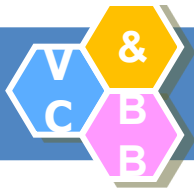
```
<tên kiểu con trỏ> <tên biến con trỏ>;
```

❖ Ví dụ

```
typedef int *pint;  
int *p1;  
pint p2, p3;
```

❖ Lưu ý khi khai báo kiểu dữ liệu mới

- Giảm bối rối khi mới tiếp xúc với con trỏ.
- Nhưng dễ nhầm lẫn với biến thường.



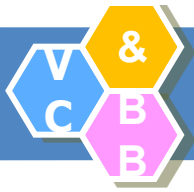
Con trỏ NULL

❖ Khái niệm

- Con trỏ **NULL** là con trỏ không trỏ vào đâu cả.
- Khác với con trỏ chưa được khởi tạo.

```
int n;  
int *p2; // unreferenced local variable  
int *p3 = NULL;
```





Khởi tạo kiểu con trỏ

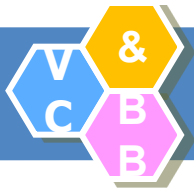
❖ Khởi tạo

- Khi mới khai báo, biến con trỏ được **đặt ở địa chỉ nào đó** (không biết trước).
 - chứa **giá trị không xác định**
- Để lấy địa chỉ của biến: **<tên_biến>**
- Đặt địa chỉ của biến vào con trỏ (toán tử **&**)

<tên biến con trỏ> = &<tên biến>;

Ví dụ

```
int a, b;  
int *pa = &a, *pb;  
pb = &b;
```

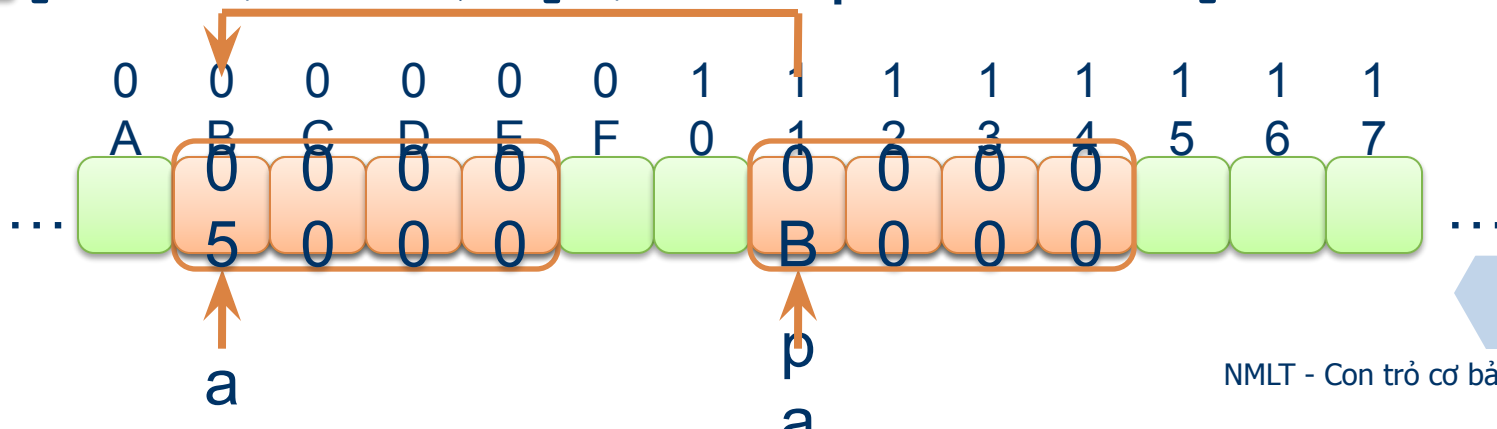


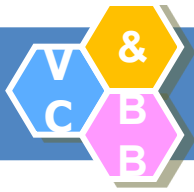
Sử dụng con trỏ

- ❖ Truy xuất đến ô nhớ mà con trỏ trỏ đến
 - Con trỏ chứa **một số nguyên chỉ địa chỉ**.
 - Vùng nhớ mà nó trỏ đến, sử dụng toán tử *****.

❖ Ví dụ

```
int a = 5, *pa = &a;  
printf("%d\n", pa); // Giá trị biến pa  
printf("%d\n", *pa); // Giá trị vùng nhớ pa trỏ đến  
printf("%d\n", &pa); // Địa chỉ biến pa
```





Kích thước của con trỏ

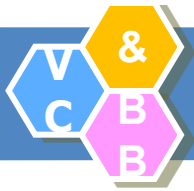
❖ Kích thước của con trỏ

```
char *p1;  
int *p2;  
float *p3;  
double *p4;
```

...

- Con trỏ **chỉ lưu địa chỉ** nên kích thước của mọi con trỏ là như nhau:
 - Môi trường MD-DOS (**16 bit**): **2** bytes
 - Môi trường Windows (**32 bit**): **4** bytes





Các cách truyền dữ liệu

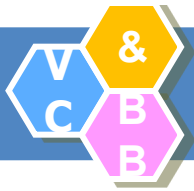
❖ Truyền giá trị (tham trị)

```
#include <stdio.h>

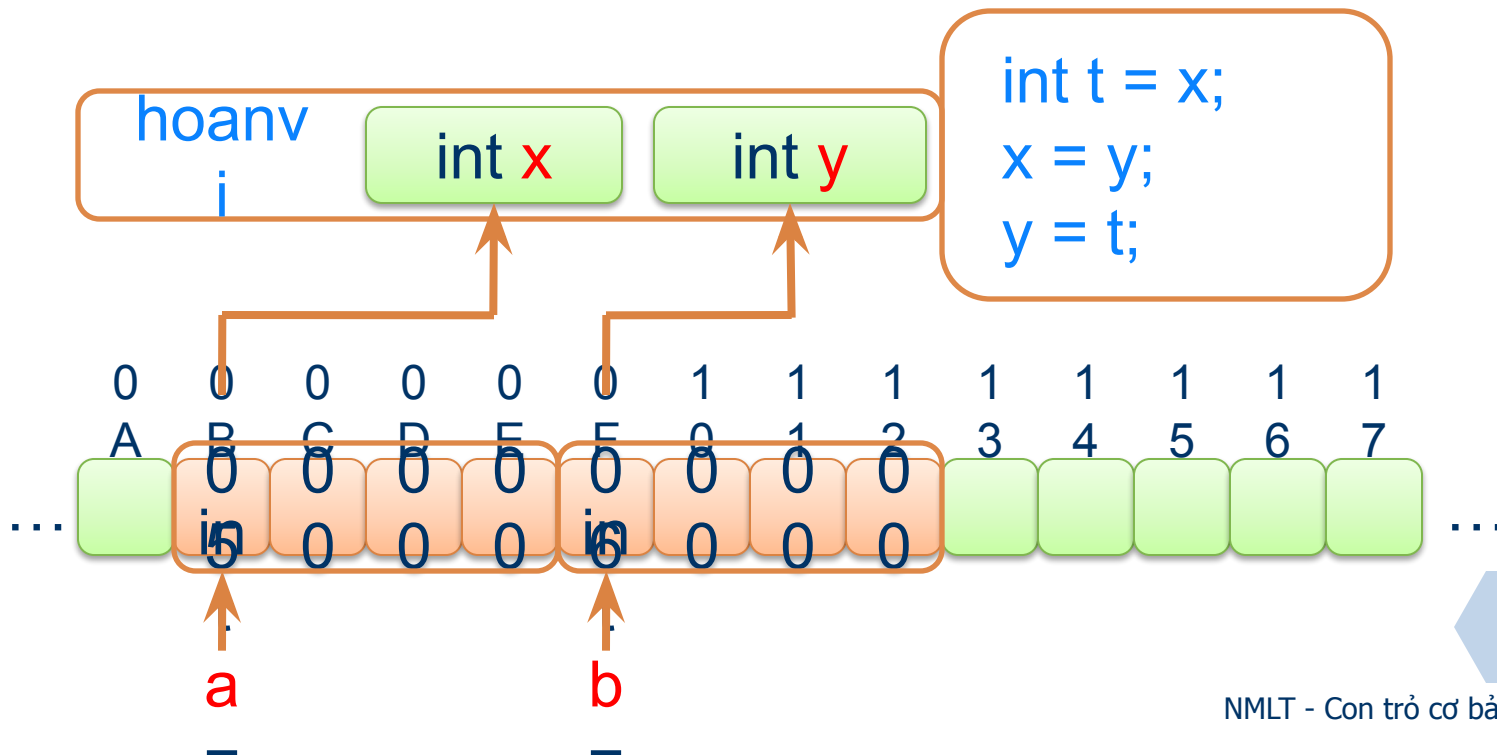
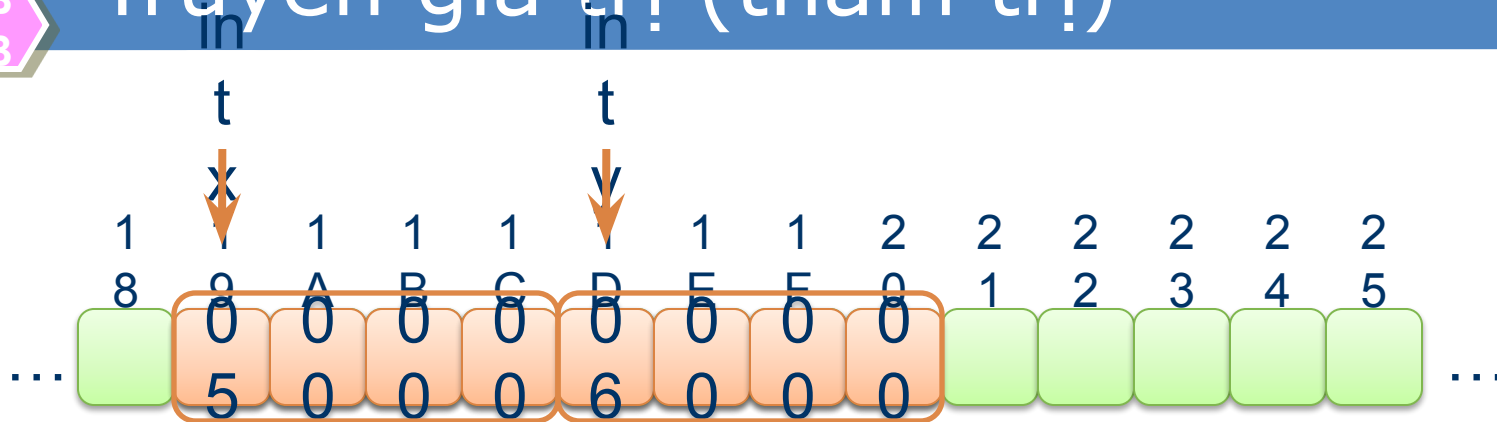
void hoanvi(int x, int y);

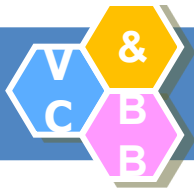
void main()
{
    int a = 5; b = 6;
    hoanvi(a, b);
    printf("a = %d, b = %d", a, b);
}

void hoanvi(int x, int y)
{
    int t = x; x = y; y = t;
}
```



Truyền giá trị (tham trị)





Các cách truyền đổi số

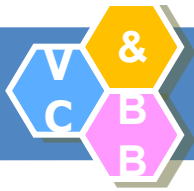
❖ Truyền địa chỉ (con trỏ)

```
#include <stdio.h>

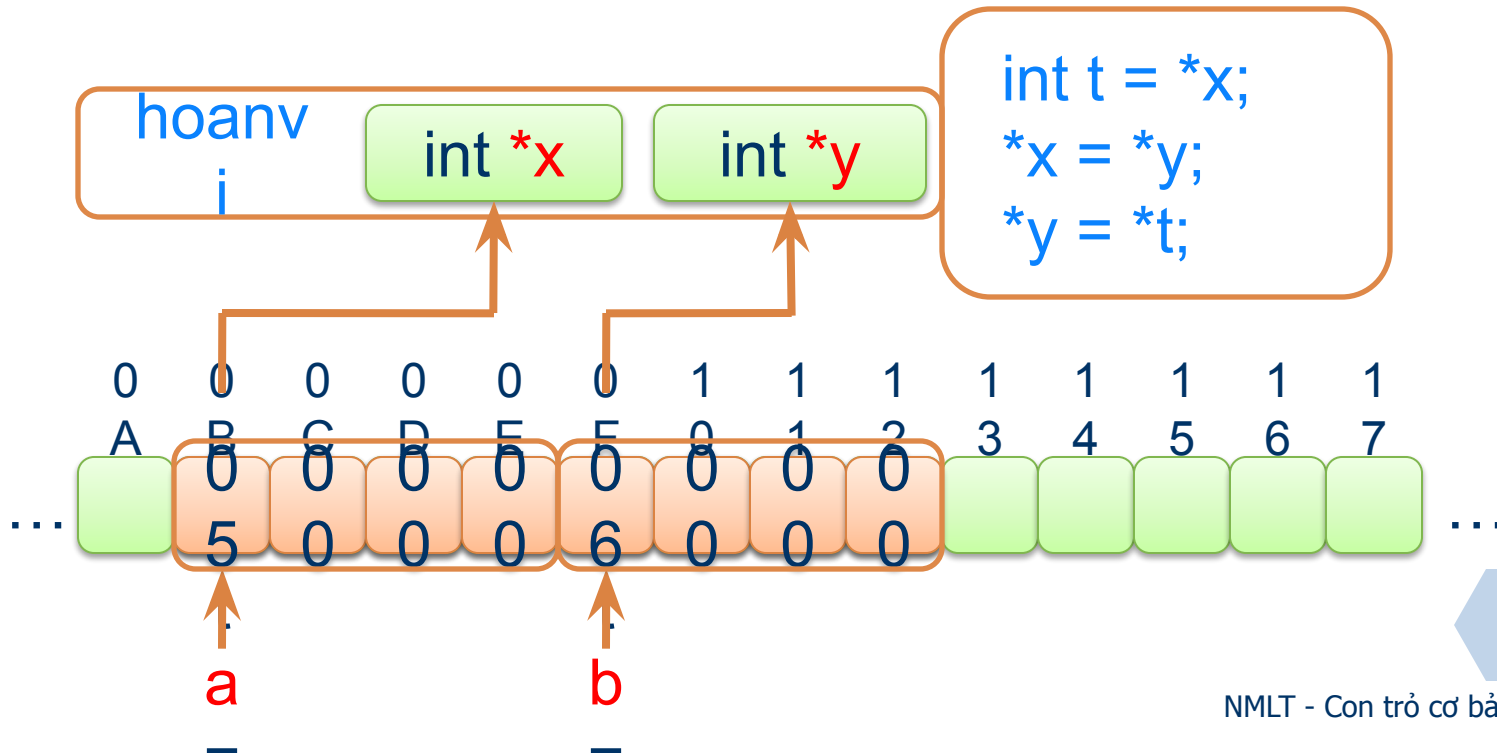
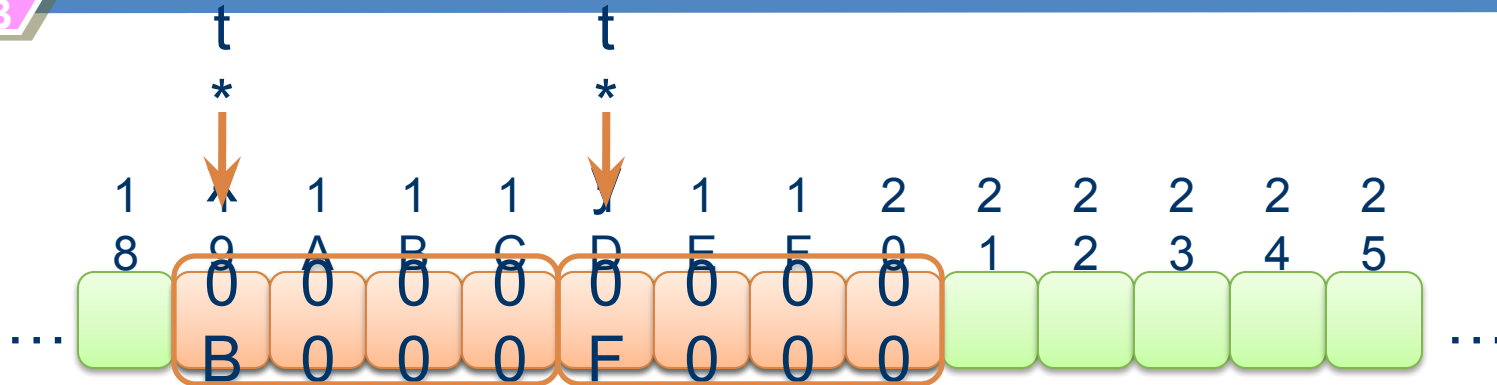
void hoanvi(int *x, int *y);

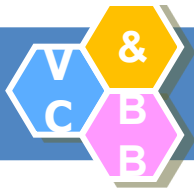
void main()
{
    int a = 2912; b = 1706;
    hoanvi(&a, &b);
    printf("a = %d, b = %d", a, b);
}

void hoanvi(int *x, int *y)
{
    int t = *x; *x = *y; *y = t;
}
```



Truyền địa chỉ (con trỏ)





Các cách truyền đổi số

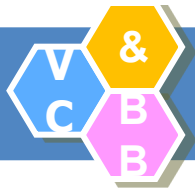
❖ Truyền tham chiếu (C++)

```
#include <stdio.h>

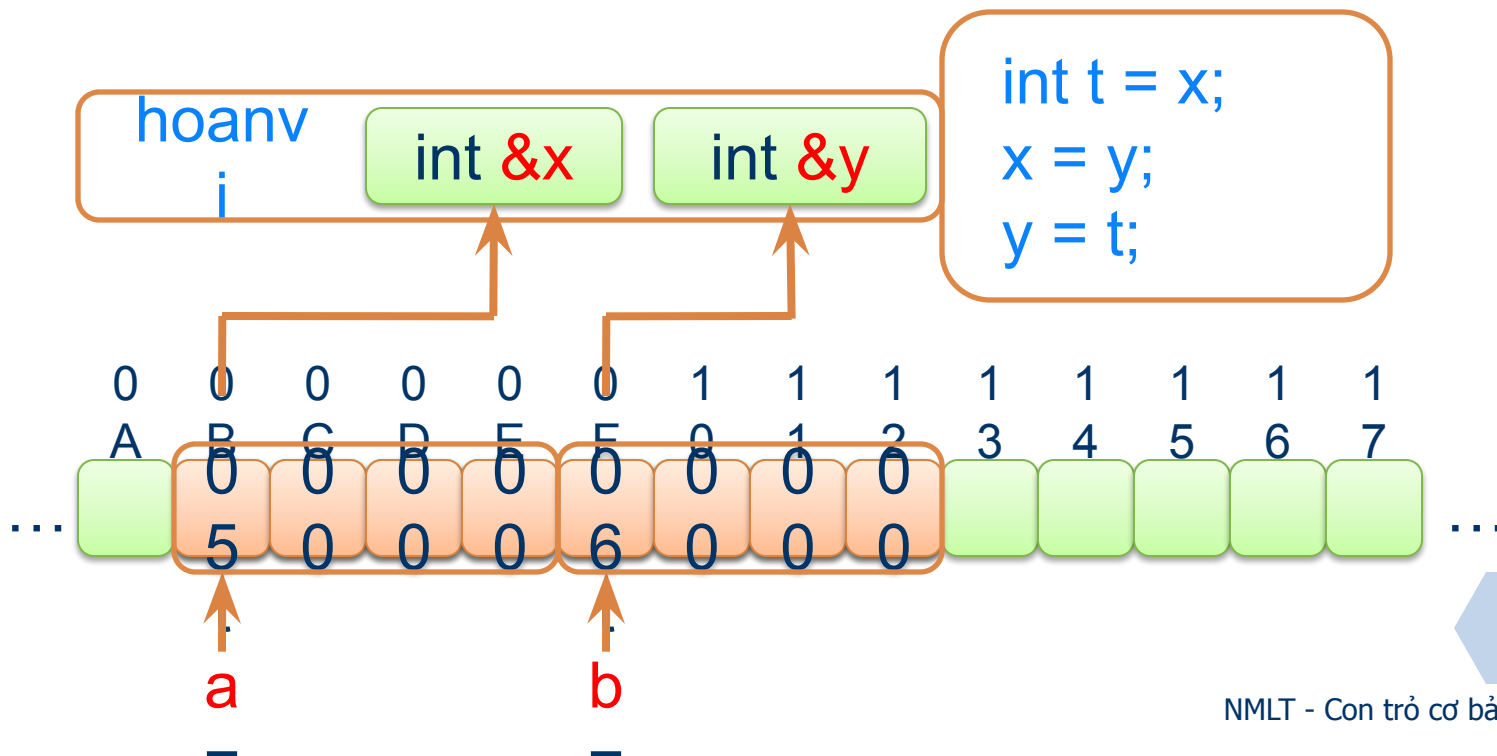
void hoanvi(int &x, int &y);

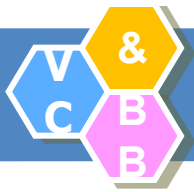
void main()
{
    int a = 2912; b = 1706;
    hoanvi(a, b);
    printf("a = %d, b = %d", a, b);
}

void hoanvi(int &x, int &y)
{
    int t = x; x = y; y = t;
}
```

Truyền tham chiếu (C++)



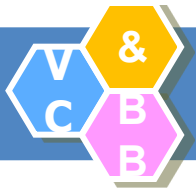


Một số lưu ý

❖ Một số lưu ý

- Con trỏ là khái niệm quan trọng và khó nhất trong C. Mức độ thành thạo C được đánh giá qua mức độ sử dụng con trỏ.
- Nắm rõ quy tắc sau, ví dụ `int a, *pa = &a;`
 - `*pa` và `a` đều chỉ **nội dung** của biến `a`.
 - `pa` và `&a` đều chỉ **địa chỉ** của biến `a`.
- **Không nên** sử dụng con trỏ khi chưa được khởi tạo. Kết quả sẽ không lường trước được.

```
int *pa; *pa = 1904;
```

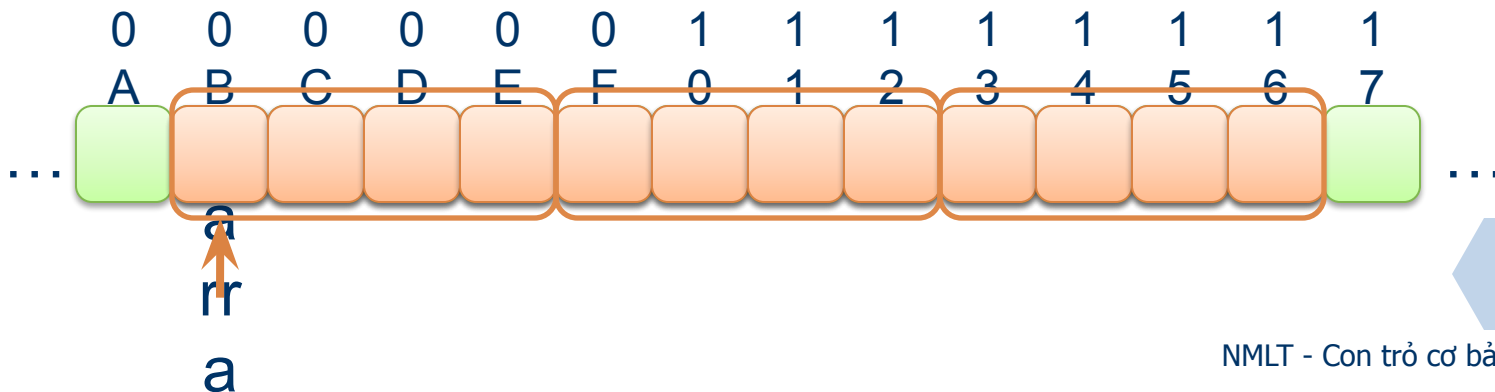


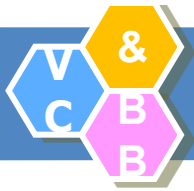
Con trỏ và mảng một chiều

❖ Mảng một chiều

```
int array[3];
```

- Tên mảng array là một **hằng con trỏ**
 - **không thể thay đổi** giá trị của hằng này.
- array là địa chỉ đầu tiên của mảng
 - **array == &array[0]**





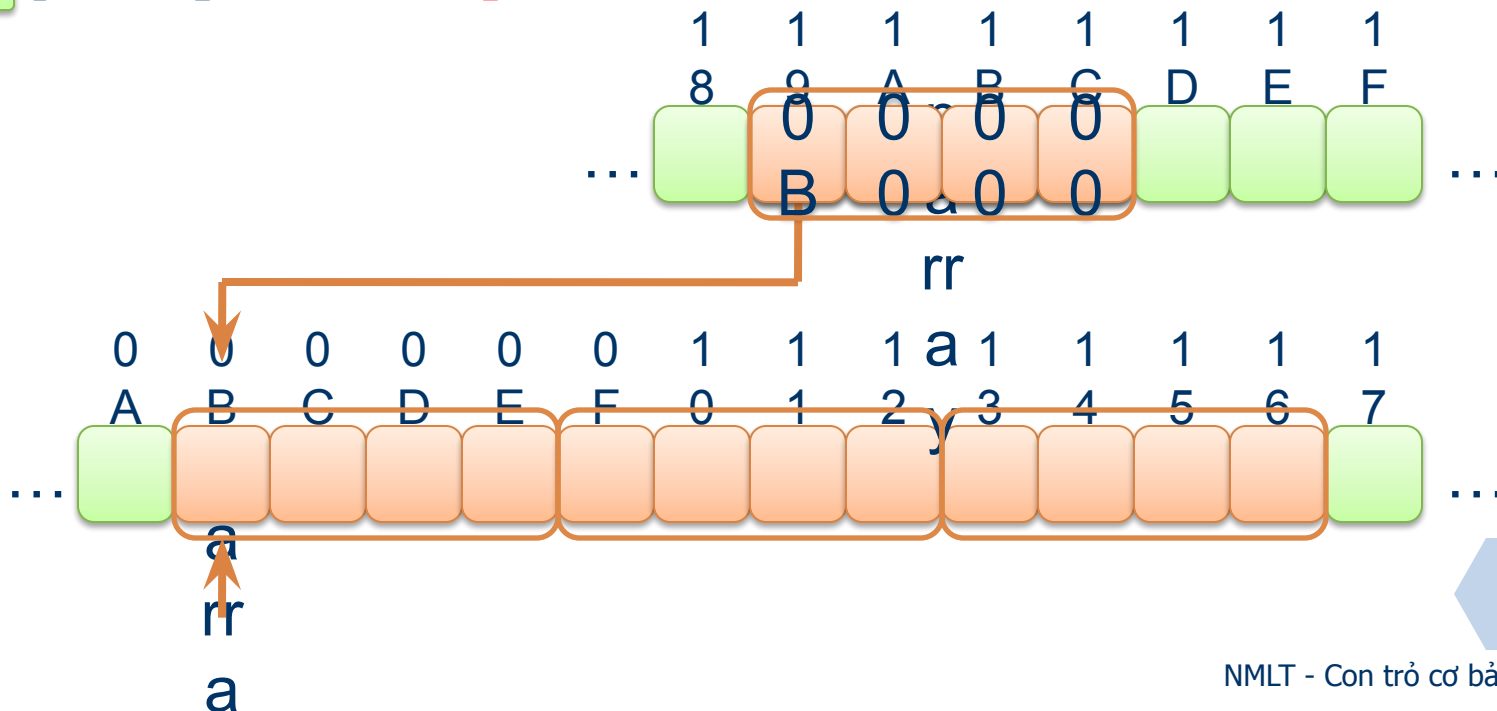
Con trỏ và mảng một chiều

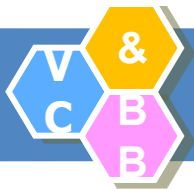
❖ Con trỏ đến mảng một chiều

```
int array[3], *parray;
```

```
parray = array; // Cách 1
```

```
parray = &array[0]; // Cách 2
```

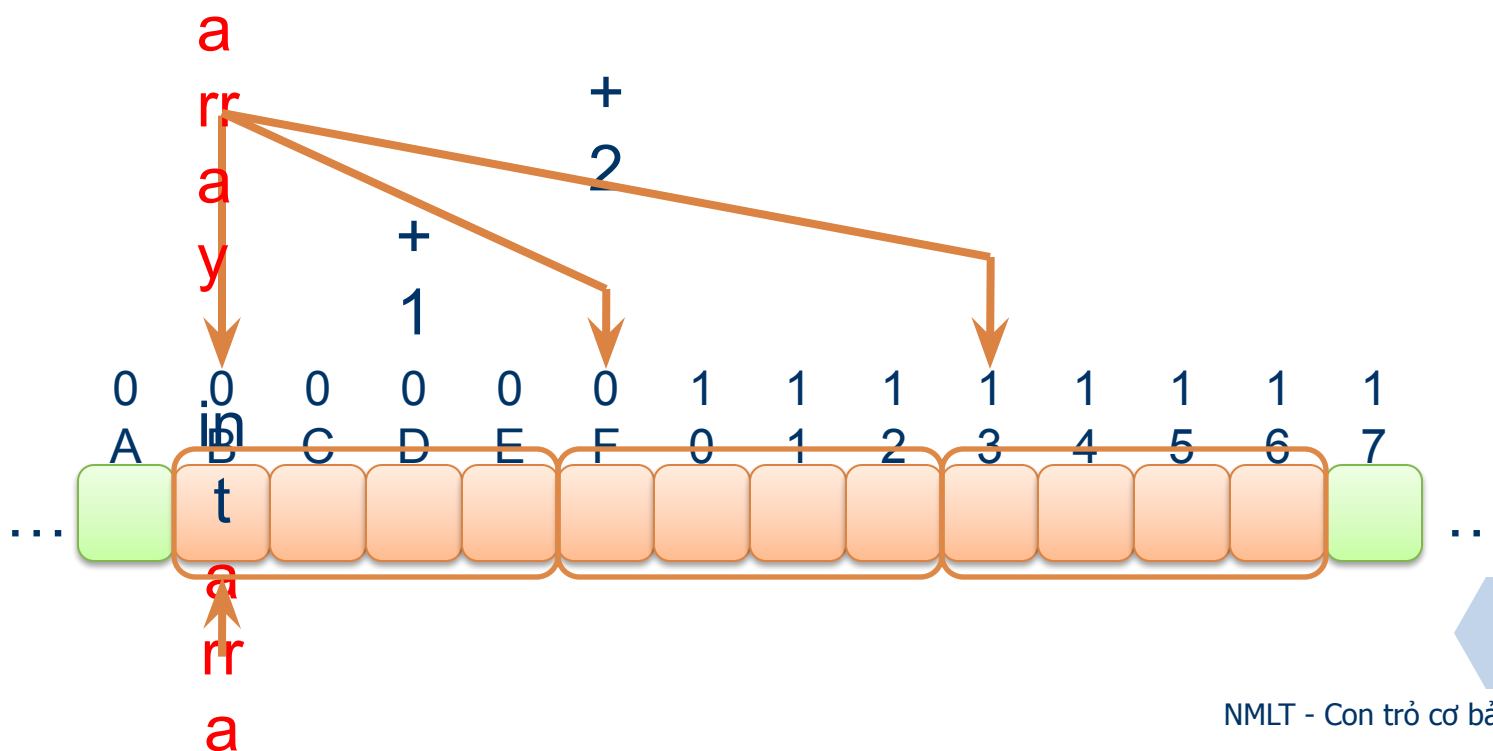




Phép toán số học trên con trỏ

❖ Phép cộng (tăng)

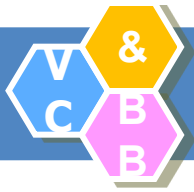
- $+ n \Leftrightarrow + n * \text{sizeof}(<\text{kiểu dữ liệu}>)$
- Có thể sử dụng toán tử gộp += hoặc ++





- $n \Leftrightarrow -n * \text{sizeof}(\text{<kiểu dữ liệu>})$
- Có thể sử dụng toán tử gộp $--$ hoặc $--$

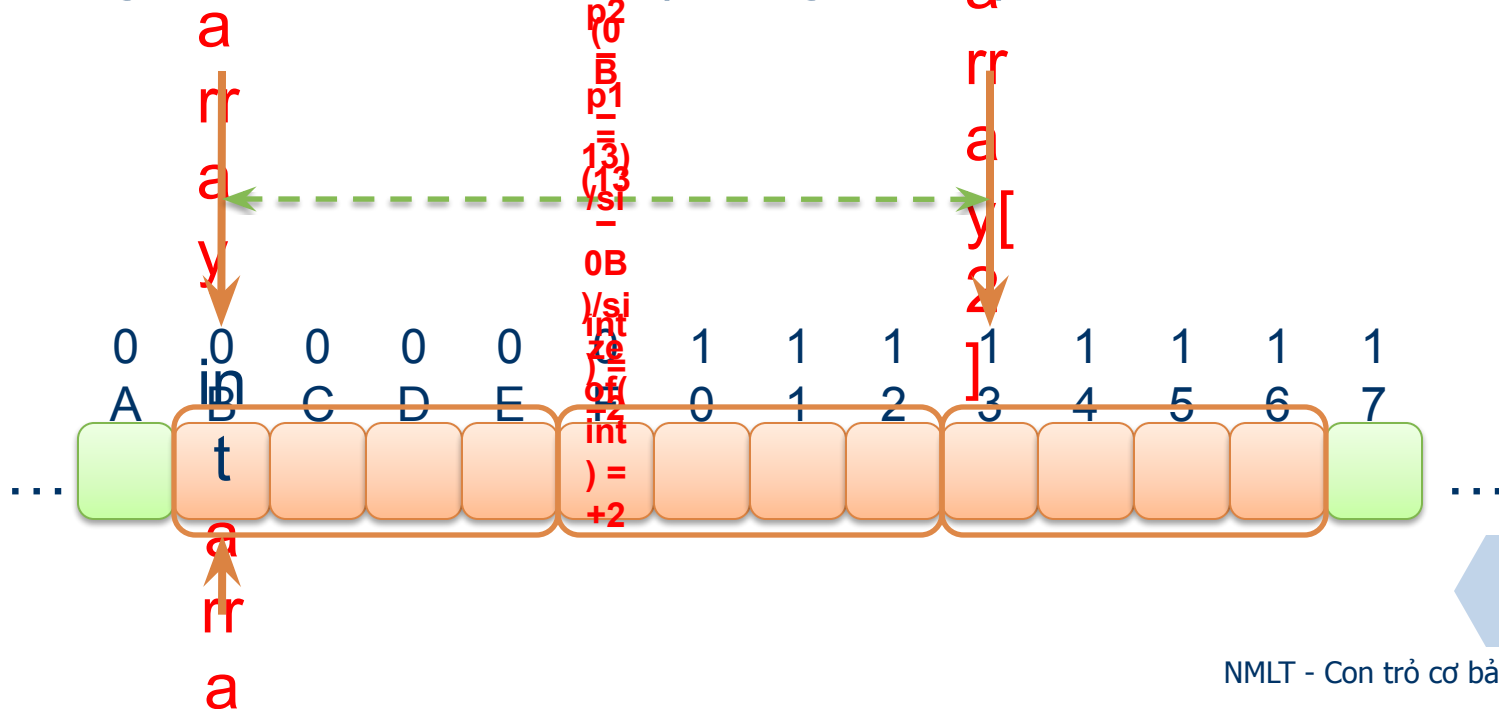


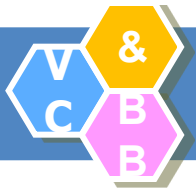


Phép toán số học trên con trỏ

❖ Phép toán tính khoảng cách giữa 2 con trỏ

- **<kiểu dữ liệu> *p1, *p2;**
- $p1 - p2$ cho ta khoảng cách (theo số phần tử) giữa hai con trỏ (cùng kiểu)

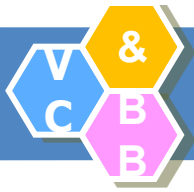




Phép toán số học trên con trỏ

❖ Các phép toán khác

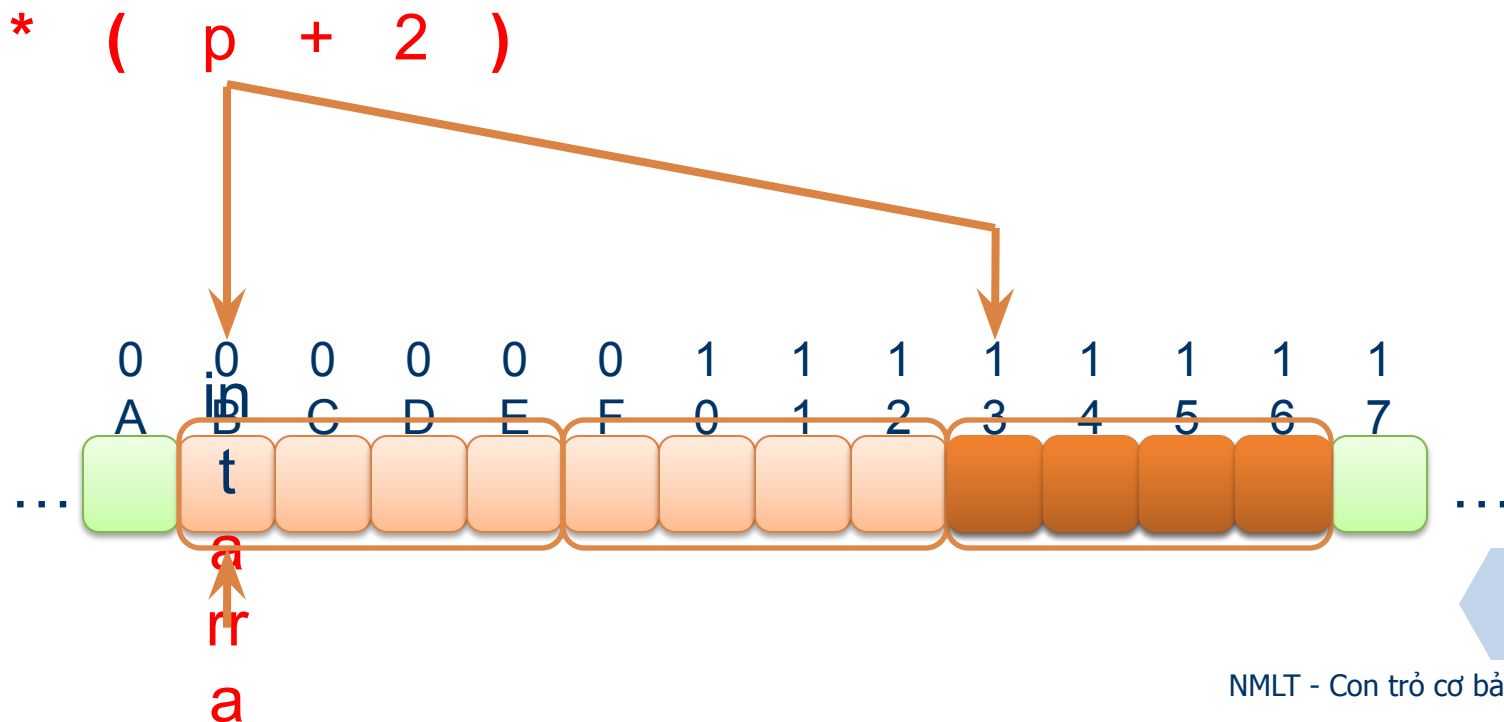
- Phép so sánh: So sánh địa chỉ giữa hai con trỏ (thứ tự ô nhớ)
 - $==$ $!=$
 - $>$ $>=$
 - $<$ $<=$
- Không thể thực hiện các phép toán: $*$ / $\%$

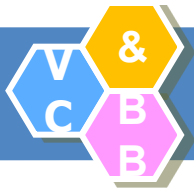


Con trỏ và mảng một chiều

❖ Truy xuất đến phần tử thứ n của mảng (không sử dụng biến mảng)

■ $\text{array}[n] == p[n] == *(p + n)$





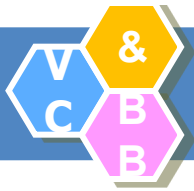
Con trỏ và mảng một chiều

❖ Ví dụ nhập mảng

```
void main()
{
    int a[10], n = 10, *pa;
    pa = a; // hoặc pa = &a[0];

    for (int i = 0; i < n; i++)
        scanf("%d", &a[i]);
        scanf("%d", &p[i]);
        scanf("%d", a + i);
        scanf("%d", p + i);
        scanf("%d", a++);
        scanf("%d", p++);
}
```

□ $\&a[i] \Leftrightarrow (a + i) \Leftrightarrow (p + i) \Leftrightarrow \&p[i]$

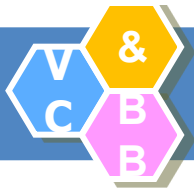


Con trỏ và mảng một chiều

❖ Ví dụ xuất mảng

```
void main()
{
    int a[10], n = 10, *pa;
    pa = a; // hoặc pa = &a[0];
    ...
    for (int i = 0; i < n; i++)
        printf("%d", a[i]);
        printf("%d", p[i]);
        printf("%d", *(a + i));
        printf("%d", *(p + i));
        printf("%d", *(a++));
        printf("%d", *(p++));
}
```

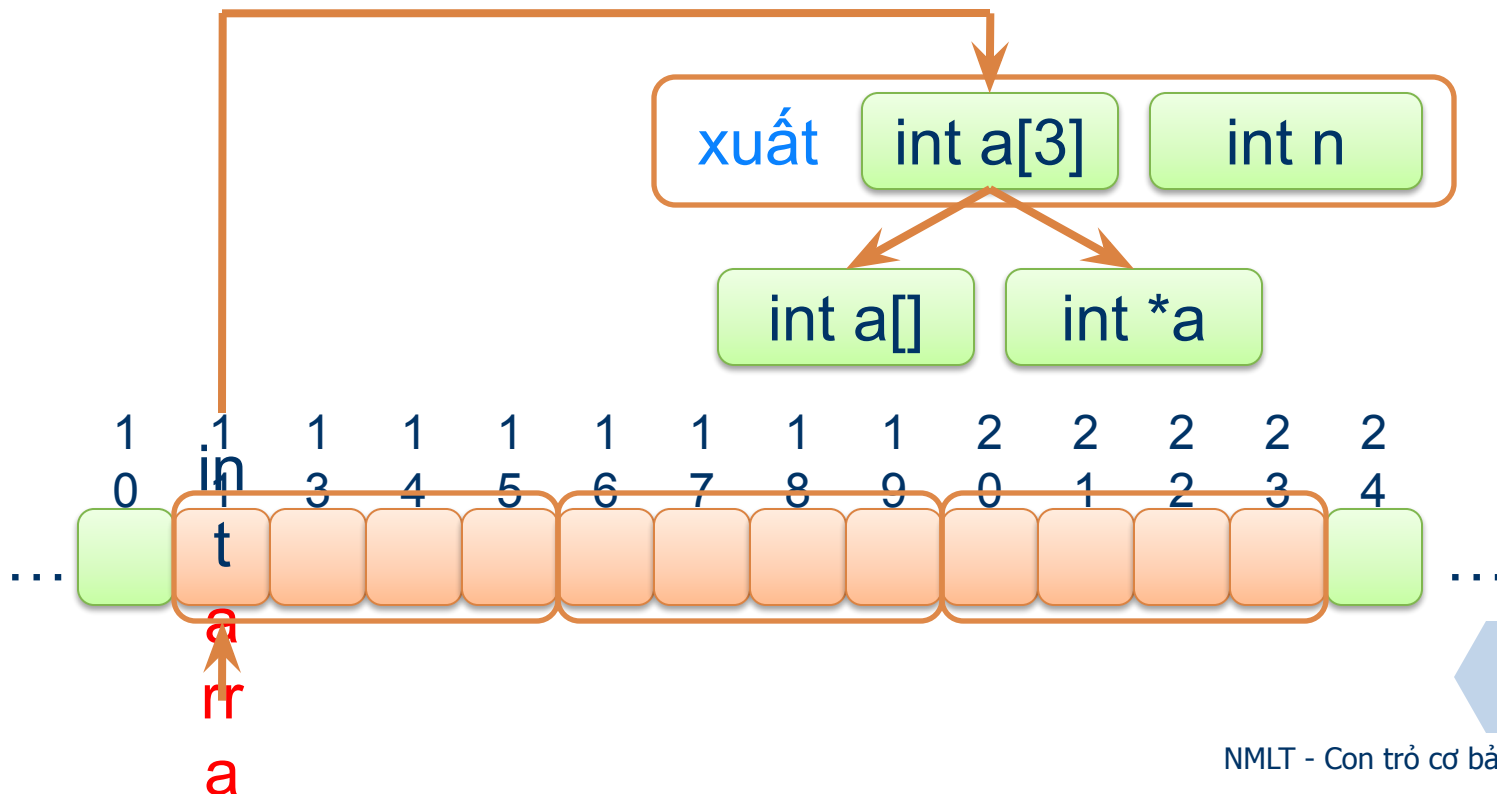
□ $a[i] \Leftrightarrow *(a + i) \Leftrightarrow *(p + i) \Leftrightarrow p[i]$

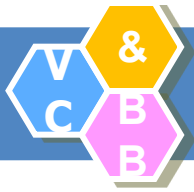


Truyền mảng 1 chiều cho hàm

❖ Chú ý!

- Mảng một chiều truyền cho hàm là **địa chỉ của phần tử đầu tiên** chứ không phải toàn mảng.





Con trỏ và mảng một chiều

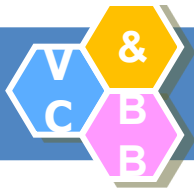
❖ Ví dụ

```
void xuat(int a[10], int n)
{
    for (int i = 0; i<n; i++)
        printf("%d", *(a++)); // OK
}

void main()
{
    int a[10], n = 10;

    for (int i = 0; i<n; i++)
        printf("%d", *(a++)); // Lỗi
}
```

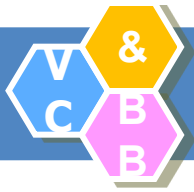
❑ Đối số mảng truyền cho hàm **không phải hằng con trỏ**.



Con trỏ và mảng một chiều

❖ Lưu ý

- Không thực hiện các phép toán nhân, chia, lấy phần dư.
- Tăng/giảm con trỏ n đơn vị có nghĩa là tăng/giảm giá trị của nó $n * \text{sizeof}(<\text{kiểu dữ liệu mà nó trỏ đến}>)$
- Không thể tăng/giảm biến mảng. Hãy gán một con trỏ đến địa chỉ đầu của mảng và tăng/giảm nó.
- Đối số mảng một chiều truyền cho hàm là địa chỉ phần tử đầu tiên của mảng.



Con trỏ cấu trúc

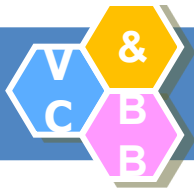
❖ Truy xuất bằng 2 cách

<tên biến con trỏ cấu trúc>-><tên thành phần>
(*<tên biến con trỏ cấu trúc>).<tên thành phần>

❖ Ví dụ

```
struct PHANSO
{
    int tu, mau;
};
PHANSO ps1, *ps2 = &p1; // ps2 là con trỏ

ps1.tu = 1; ps1.mau = 2;
ps2->tu = 1; ps2->mau = 2;
(*ps2).tu = 1; (*ps2).mau = 2;
```



Con trỏ cấu trúc

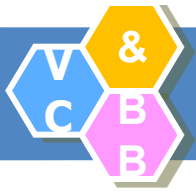
❖ Gán hai cấu trúc

```
struct PHANSO
{
    int tu, mau;
};
PHANSO ps1, *ps2;
```

```
ps1.tu = 1; ps1.mau = 2; // ps1 = 1/2
```

```
ps2 = &ps1;
```

```
ps2->tu = 3; ps2->mau = 4; // ps1 = 3/4
```

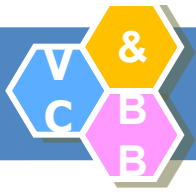
Bài tập lý thuyết

❖ Bài 1: Cho đoạn chương trình sau:

```
float pay;  
float *ptr_pay;  
pay=2313.54;  
ptr_pay = &pay;
```

❖ Hãy cho biết giá trị của:

- a. pay
- b. *ptr_pay
- c. *pay
- d. &pay



Bài tập lý thuyết

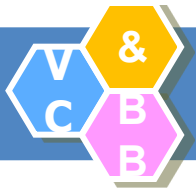
❖ Bài 2: Tìm lỗi

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

void main()
{
    int *x, y = 2;

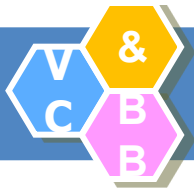
    *x = y;
    *x += y++;

    printf("%d %d", *x, y) ;
    getch() ;
}
```



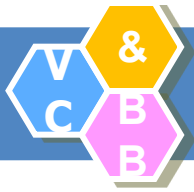
Bài tập lý thuyết

- ❖ Bài 1: Toán tử nào dùng để xác định địa chỉ của một biến?
- ❖ Bài 2: Toán tử nào dùng để xác định giá trị của biến do con trỏ trỏ đến?
- ❖ Bài 3: Phép lấy giá trị gián tiếp là gì?
- ❖ Bài 4: Các phần tử trong mảng được sắp xếp trong bộ nhớ như thế nào?
- ❖ Bài 5: Cho mảng một chiều data. Trình bày 2 cách lấy địa chỉ phần tử đầu tiên của mảng này.



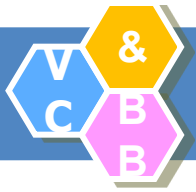
Bài tập lý thuyết

- ❖ **Bài 6:** Nếu ta truyền cho hàm đối số là mảng một chiều. Trình bày hai cách nhận biết phần tử cuối của mảng?
- ❖ **Bài 7:** Trình bày 6 phép toán có thể thực hiện trên con trỏ?
- ❖ **Bài 8:** Cho con trỏ p1 trỏ đến phần tử thứ 3 còn con trỏ p2 trỏ đến phần tử thứ 4 của mảng int.
 $p2 - p1 = ?$
- ❖ **Bài 9:** Giống như câu trên nhưng đối với mảng float?



Bài tập

- ❖ Bài 10: Trình bày khai báo con trỏ pchar trở đến kiểu char.
- ❖ Bài 11: Cho biến cost kiểu int. Khai báo và khởi tạo con trỏ pcost trở đến biến này.
- ❖ Bài 12: Gán giá trị 100 cho biến cost sử dụng hai cách trực tiếp và gián tiếp.
- ❖ Bài 13: In giá trị của con trỏ và giá trị của biến mà nó trỏ tới.
- ❖ Bài 14: Sử dụng con trỏ để làm lại các bài tập về mảng một chiều.



Bài tập lý thuyết

❖ Bài 15: Cho đoạn chương trình sau:

```
int *pint;
```

```
float a;
```

```
char c;
```

```
double *pd;
```

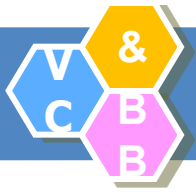
Hãy chọn phát biểu sai cú pháp:

a. `a = *pint;`

b. `c = *pd;`

c. `*pint = *pd;`

d. `pd = a;`



Bài tập thực hành

- ❖ **Bài 16:** Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$), sắp xếp các chữ số của n theo thứ tự tăng dần.

Ví dụ:

- Nhập $n = 1536$
- Kết quả sau khi sắp xếp: 1356.