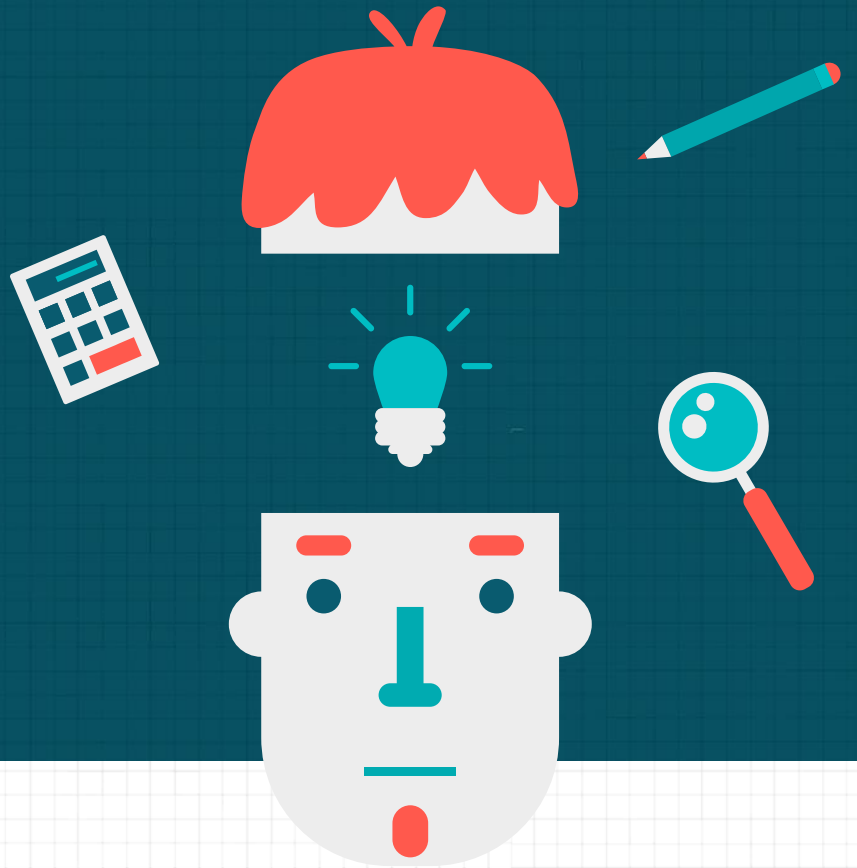
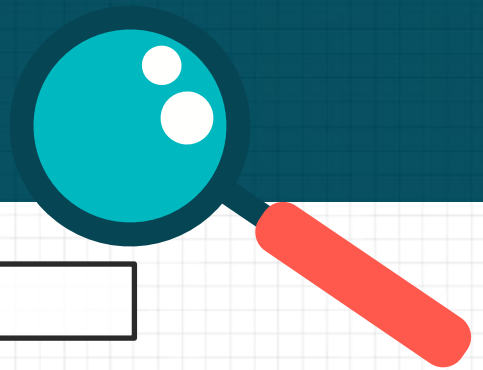
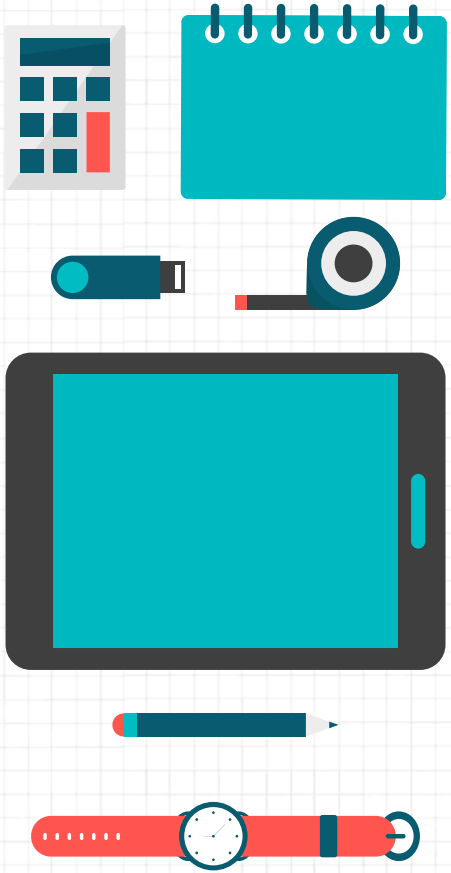
Bài 1
Con trỏ



Ths. Phạm Minh Hoàng

Nội dung



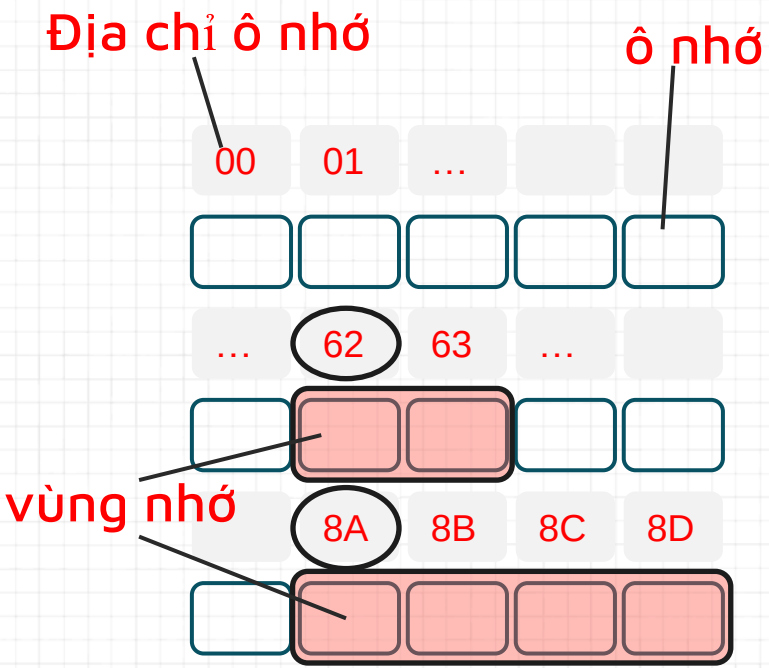


01

Các khái niệm cơ bản về con trỏ

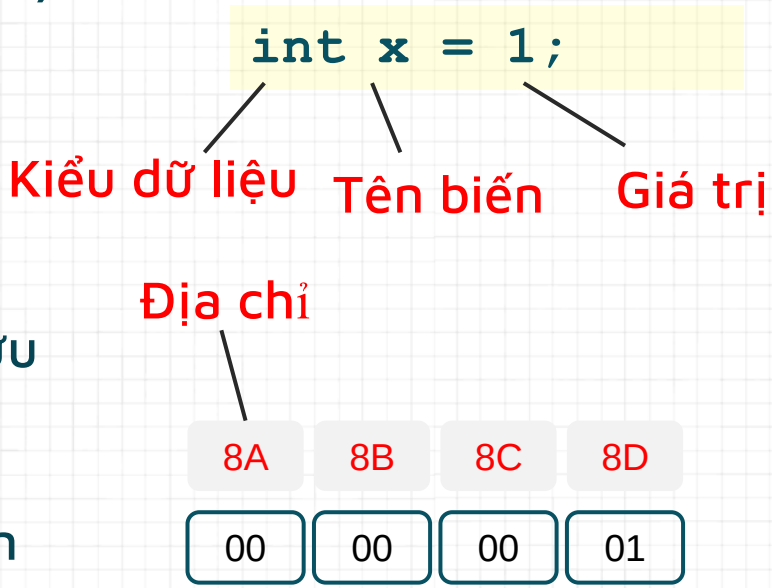
Địa chỉ vùng nhớ

- Bộ nhớ dùng để chứa
  - Hệ điều hành
  - Chương trình: các câu lệnh thực thi
  - Dữ liệu
- Bộ nhớ được chia thành các ô nhớ. Mỗi ô nhớ có kích thước 1 byte.
- Mỗi ô nhớ được đánh bằng địa chỉ ô nhớ.
- Để lưu trọn vẹn dữ liêu cần nhiều ô nhớ nằm liên tiếp nhau tạo thành vùng nhớ.
  - Vùng nhớ số nguyên short: 2 byte
  - Vùng nhớ số nguyên int: 4 byte
- Địa chỉ vùng nhớ là địa chỉ của ô nhớ đầu tiên



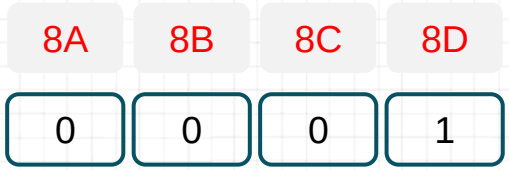
Biến

- Biến (variable) là sự ảo hóa (abstraction) của một vùng nhớ lưu dữ liệu
- Thuộc tính của biến
  - Tên (name): định danh cho vùng nhớ
  - Kiểu dữ liệu (data type): xác định loại biến và kích thước vùng nhớ lưu biến đó
  - Giá trị (value): dữ liệu được lưu trong vùng nhớ tương ứng của biến
  - Địa chỉ (address): địa chỉ vùng nhớ của biến



Toán tử lấy địa chỉ

- Toán tử &
  - Cú pháp `&<tên biến>;`
  - Input: biến cần lấy địa chỉ
  - Output: địa chỉ vùng nhớ của biến



Ví dụ

```
int a = 1;
cout << &a;
```

Con trỏ

Định nghĩa

Con trỏ là biến lưu địa chỉ của một vùng nhớ hợp lệ khác

- Khai báo con trỏ
  - Cú pháp

```
<type>* <name>;
```

  - <name>: tên biến con trỏ
  - <type>: kiểu dữ liệu của vùng nhớ mà con trỏ giữ địa chỉ
- Ví dụ

```
int* p;
double *q;
char * r;
```

Gán giá trị cho con trỏ

- Gán giá trị mặc định

```
int* p = NULL;
double* q = nullptr;
char* r = 0x01;
```

- Gán địa chỉ vùng nhớ của biến khác

```
int a = 1;
int* p = &a;
```

- Con trỏ chỉ có thể giữ địa chỉ vùng nhớ phù hợp kiểu đã khai báo

```
int a = 1;
int* p = &a; //yes
float b = 3.14;
int* q = &b; //no
```

Gán giá trị cho con trỏ

- Có thể ép kiểu con trỏ từ kiểu dữ liệu này sang kiểu dữ liệu khác

```
int a = 1;
int* p = &a;
float b = 3.14;
int* q = (int*)&b;
```

- Gán con trỏ cho con trỏ khác → hai con trỏ lưu cùng địa chỉ của một vùng nhớ

```
int a = 10;
int* p = &a;
int* q = p;
```

Đặc điểm con trỏ

```
int a = 1
int* p = &a;
```

Con trỏ p lưu địa chỉ vùng nhớ của biến a

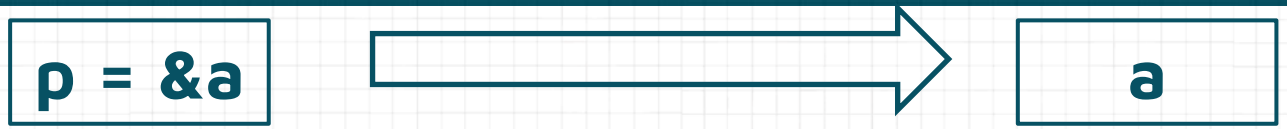
Con trỏ p trỏ đến vùng nhớ của biến a

Con trỏ p trỏ đến biến a

Đặc điểm con trỏ

- Con trỏ cũng được cấp phát vùng nhớ
- Kích thước vùng nhớ của con trỏ là cố định
- Kích thước vùng nhớ của con trỏ bằng kích thước kiểu số nguyên
- Giá trị lưu trong vùng nhớ của con trỏ là địa chỉ của vùng nhớ khác

Toán tử truy xuất nội dung vùng nhớ



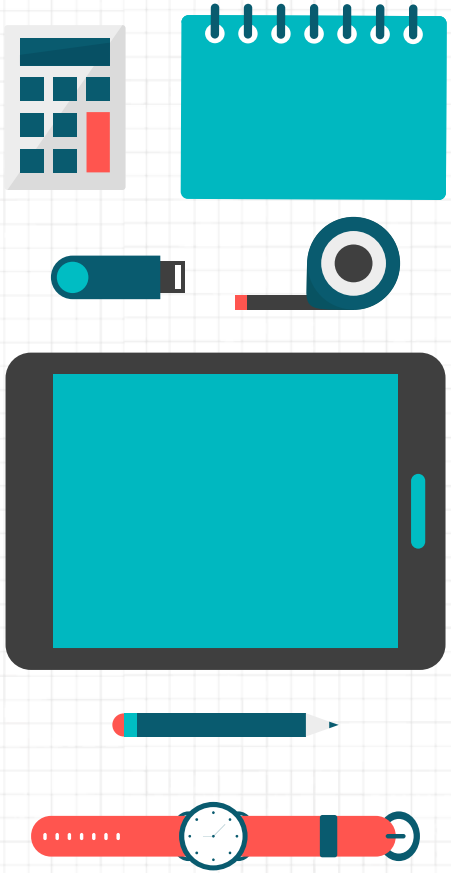
- Toán tử *
  - Cú pháp `*<pointer>;`
  - Input: biến con trỏ
  - Output: giá trị chứa trong vùng nhớ con trỏ đang trỏ đến
- Ví dụ

```
int a = 1;
int* p = &a;
int b = *p;
cout  << b;
```

Bài tập

```
int a = 1;
int b;
int* p = &a;
int* q = &b;
*p = 10;
b = 20;
cout  << *p << *q;
```

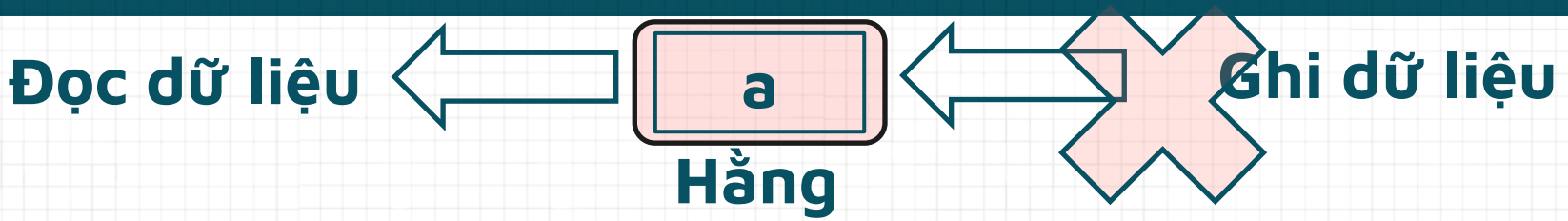
```
int a = 1;
int b = 20;
int* p = &a;
int* q = &b;
*p = *q;
cout  << a << b;
q = p;
*q = 100;
cout  << a << b;
```



02

Con trỏ và hằng

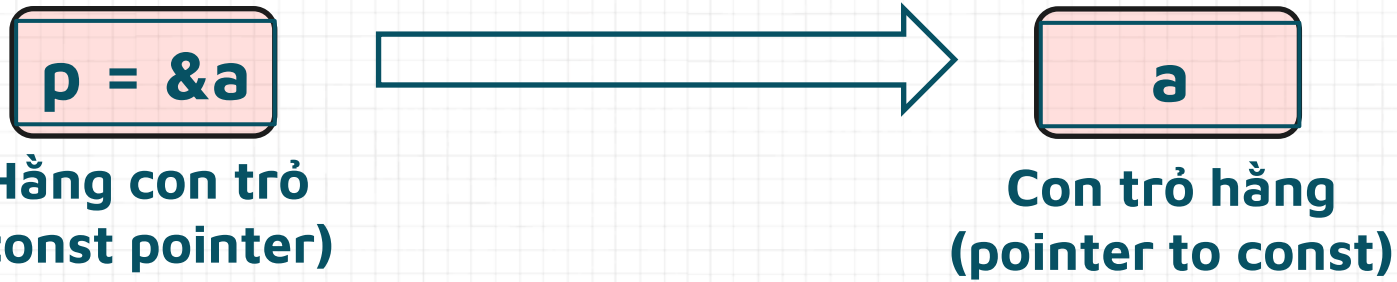
Hằng



- Hằng là sự ảo hóa vùng nhớ chỉ cho phép đọc dữ liệu, không cho phép ghi dữ liệu
  - Cú pháp

```
<type> const <name>;
const <type> <name>;
```

  - Chữ const đứng trước cái gì, thứ đó biến thành hằng



Con trỏ hằng

- Con trỏ hằng (pointer to const): vùng nhớ con trỏ đang trỏ đến là vùng nhớ hằng (read-only)

```
int a = 1;
const int* p = &a;
*p = 10; //error
```

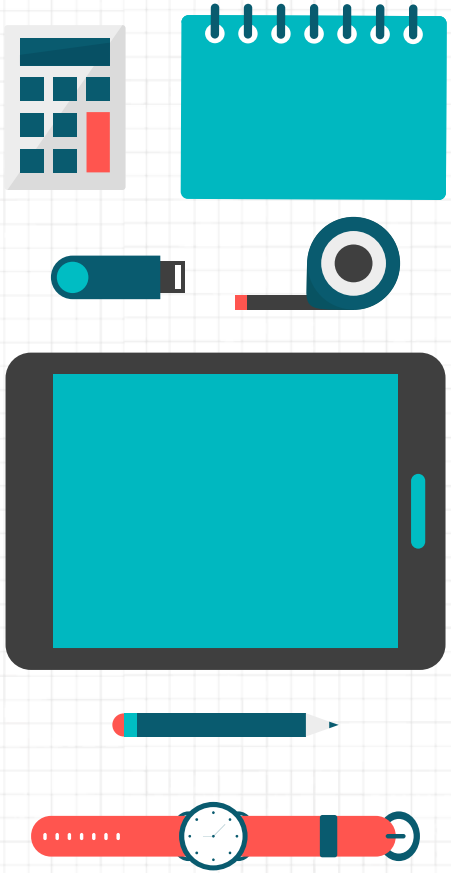
- Lưu ý: chỉ khi truy xuất đến vùng nhớ của con trỏ hằng thông qua chính con trỏ đó thì mới không hợp lệ

```
int a = 1;
const int* p = &a;
*p = 10; //error
a = 10; //ok
```

Hằng con trỏ

- Hằng con trỏ (const pointer): vùng nhớ dành để lưu chính con trỏ đó là vùng nhớ hằng

```
int* const p = 0x01;
p = 0x02; //error
```



03

Con trỏ và mảng

Mảng tĩnh

```
int a[3]={1, 2, 3};
```

&a ~ &a[0] ~ a = 00 00 00 01

- Cấp phát 3 phần tử nằm liên tiếp nhau, mỗi phần tử là 1 vùng nhớ có kích thước số nguyên
- Biến mảng a lưu địa chỉ phần tử đầu tiên a[0] của mảng
- Biến mảng a không thể thay đổi trong suốt thời gian tồn tại
- Liệu mảng có phải là một hằng con trỏ?

KHÔNG

Mảng tĩnh

- Mảng tĩnh là cấu trúc dữ liệu đặc biệt, tập hợp nhiều phần tử cùng kiểu dữ liệu, có thự tự, và nằm liên tiếp nhau trên vùng nhớ có kích thước cố định.
- Biến mảng có tính chất giống con trỏ là lưu địa chỉ phần tử đầu tiên, nhưng nó không phải là con trỏ
  - Có thể gán biến mảng cho một con trỏ

```
int a[3] = {1, 2, 3};
int* p = a;
```

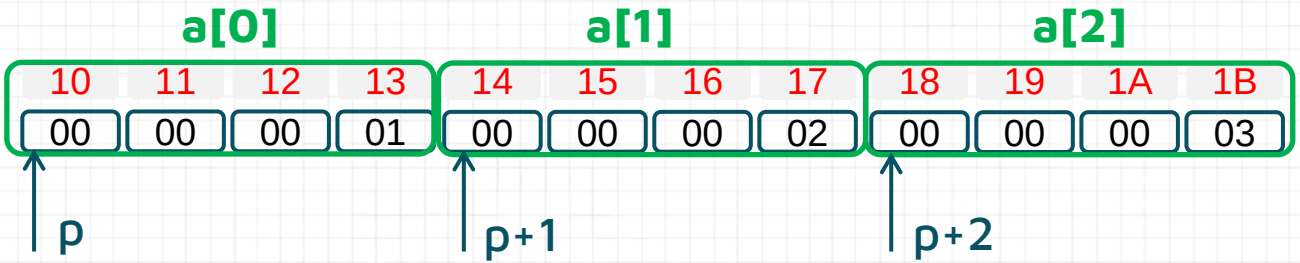
  - Không thể làm ngược lại, gán biến nào khác cho một biến mảng

```
int a[3] = {1, 2, 3};
int* p = nullptr;
a = p; //error
```

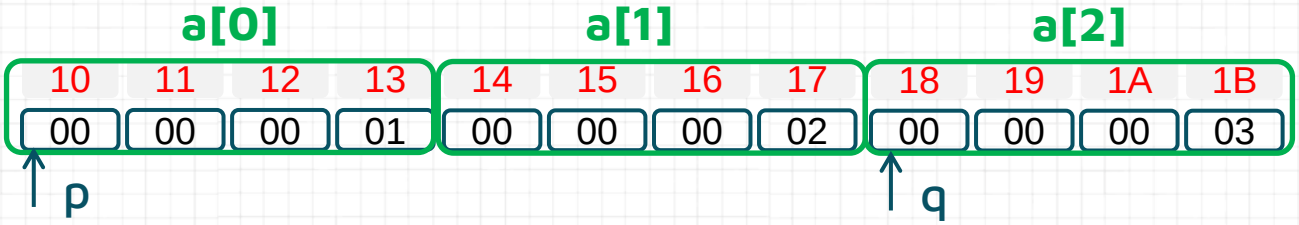
Di chuyển con trỏ

- Di chuyển con trỏ về trước/sau trên các vùng nhớ nằm liên tiếp nhau
- Toán tử: +/-, ++, --, +=, -=
- Độ lớn khoảng di chuyển dựa vào kích thước kiểu dữ liệu biến con trỏ
- Công thức <pointer> +/- k = <address> +/- k*sizeof(<type>)

```
int a[3] = {1, 2, 3};
int* p = a;
cout << (p + 1);
cout << *(p + 2);
```



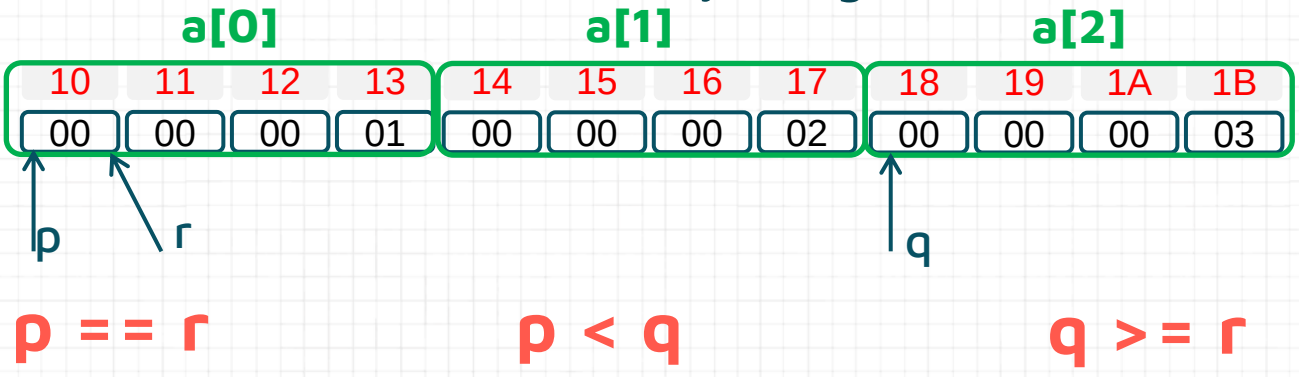
Khoảng cách giữa hai con trỏ



- Khoảng cách giữa hai con trỏ p và q cùng kiểu tính bằng p – q hoặc q – p.
- p – q = (10 – 18)/sizeof(int) = -2
- q – p = (18 – 10)/sizeof(int) = 2

So sánh giữa hai con trỏ

- So sánh giữa hai con trỏ dựa vào địa chỉ vùng nhớ mà chúng đang trỏ tới (thứ tự ô nhớ).
  - So sánh khác: !=
  - So sánh bằng: ==
  - So sánh lớn (lớn hơn hay bằng): > và >=
  - So sánh nhỏ (nhỏ hơn hay bằng): < và <=



Truy xuất phần tử mảng

```
int a[3] = {1, 2, 3};
int* p = a;
```

- Truy xuất phần tử có chỉ số i trong mảng\
  - Toán tử []
    - Biến mảng: a[i]
    - Con trỏ: p[i]
  - Toán tử *:
    - Biến mảng: *(a+i)
    - Con trỏ: *(p+i)

a[i] ~ *(a+i) ~ p[i] ~ *(p+i)

Truy xuất phần tử mảng

- Ví dụ: cho biết đoạn chương trình xuất ra kết quả gì?

```
int a[3] = {1, 2, 3};
int* p = a;
cout << *p;
*(p + 1) = 10;
cout << p[1];
p += 2;
p[-1] = 20;
cout << *(p - 1);
```

Bài tập

Tìm dòng bị lỗi trong đoạn code sau

```
int a[6] = {1, 2, 3, 4, 5, 6};
int* p = a;
for(int i = 0; i < 6; i++){
	cout << a[i];
	cout << p[i];
	cout << *(a + i);
	cout << *(p + i);
	cout << *(a++);
	cout << *(p++);
}
```

Bài tập

Tìm dòng bị lỗi trong đoạn code sau

```
const char* ptr;
char str[20] = “Hello world!”;
str[0] = ‘t’;
ptr = str;
ptr[1] = ‘A’;
*(char*)ptr = ‘a’;
```

Bài tập

Kết quả đoạn code khi thay dòng … lần lượt bằng các dòng lệnh trong bảng

```
int a[3] = {5, 10, 15};
int* p = &a[0];
int* q = &a[1];
…
cout << a[0] << a[1] << a[2];
cout << *p << *q;
```

1	`*p++ = 100;`
2	`*++p = 100;`
3	`++*p;`
4	`(*p)++;`
5	`*p++ = *q++;`

Bảng độ ưu tiên

1	::	Trái->phải
2	() [] . -> a++ a--	Trái->phải
3	++a --a +a -a ! ~ *a &a	Phải->trái
4	* /	Trái->phải
5	+ -	Trái->phải

- http://users.ece.utexas.edu/~ryerraballi/CPrimer/CDeclPrimer.html

Con trỏ đến vùng nhớ có kích thước cố định

- Con trỏ p trỏ đến vùng nhớ có kích thước là một số nguyên int (4 byte)

```
int a = 1;
int* p = &a;
```

  - Làm thế nào để con trỏ p trỏ đến vùng nhớ có kích thước bằng đúng 4 số nguyên int (16 byte)
  - Tổng quát: khai báo con trỏ p trỏ đến vùng nhớ có kích thước bằng đúng N số nguyên int (4N byte) với N là hằng
  - Cú pháp

```
<type> <array_name>[N];
<type> (*pointer)[N] = &<array_name>;
```

Con trỏ đến vùng nhớ có kích thước cố định

- Ví dụ

```
int a[10] = {1};
int (*p)[10] = &a;
```

- Truy xuất đến từng phần tử của mảng thông qua con trỏ p

```
cout << (*p)[1];
```