

# Ôn tập về Con trỏ

Lập trình hướng đối tượng

## Ôn tập con trỏ

- n Cấp phát động (Dynamic Allocation)  
**new, delete**
- n Con trỏ lạc (Dangling pointers)
- n Rò rỉ bộ nhớ (Memory leakage)
- n Con trỏ mảng (Array Pointer)
- n Các phép tính trên con trỏ (Pointer Arithmetic)
- n Con trỏ tới bản ghi
- n Cấp phát động mảng

# Con trỏ

```
int x = 361;
int *y = &x;
```



Một con trỏ hay một biến con trỏ là:

- .. một biến chiếu đến một ô nhớ.
- .. nó lưu vị trí/địa chỉ của ô nhớ đó.

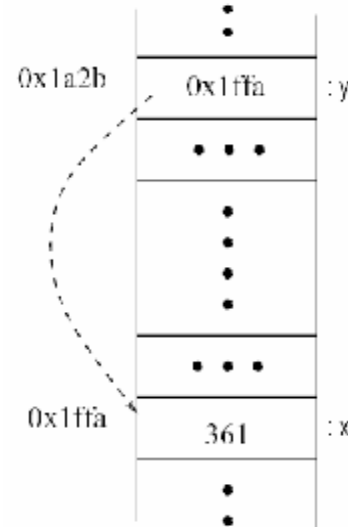
n Hai ứng dụng chính:

- .. Truy nhập gián tiếp
- .. Bộ nhớ động

n Vấn đề kỹ thuật:

Nếu P là một biến con trỏ

- .. Làm thế nào để trỏ P đến một ô nhớ nào đó?
- .. Làm thế nào để truy nhập đến ô nhớ P trỏ đến?



# Thao tác con trỏ

n Các ký hiệu, từ khóa: **&**, **\***, **new**, **delete**

```
int X, Y;
int* P;    // P is an integer pointer variable
```

n Lệnh thứ hai khai báo một biến con trỏ P có giá trị chưa xác định nhưng khác Null. Biến con trỏ này có thể chỉ trỏ tới một ô nhớ chứa một số nguyên

```
P = &Y;    // trỏ P tới Y (P lưu địa chỉ của Y)
*P = X;    // ghi giá trị của biến X vào vùng bộ nhớ trỏ bởi P
```

n Ví dụ

```
Y = 5;      // variable Y stores value
P = &X;     // P points to memory location of X
*P = Y;     // same as writing X = Y
```

Sau ví dụ trên, X = 5, Y = 5, và P trỏ tới X

# Ví dụ

```
#include <iostream>
int main()
{   int x = 10; int y = 20;
    int *p1, *p2;

    p1 = &x;
    p2 = &y;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "*p1 = " << *p1 << endl;
    cout << "*p2 = " << *p2 << endl << endl;

    *p1 = 50;
    *p2 = 90;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "*p1 = " << *p1 << endl;
    cout << "*p2 = " << *p2 << endl << endl;

    p1 = p2;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "*p1 = " << *p1 << endl;
    cout << "*p2 = " << *p2 << endl << endl;
}
```

```
x = 10
y = 20
*p1 = 10
*p2 = 20
```

```
x = 50
y = 90
*p1 = 50
*p2 = 90
```

```
x = 50
y = 90
*p1 = 90
*p2 = 90
```

# Ký hiệu

- n Đọc **\*P** là **biến mà P trỏ tới**
- n Đọc **&x** là **địa chỉ của X**
- n **&** là **toán tử địa chỉ (address of operator)**
- n **\*** là **toán tử tham nhập (dereferencing operator)**
- n Giả sử **p1 = &x** và **p2 = &y**, thì **p1** trỏ tới **x** và **p2** trỏ tới **y**

**p1 = p2**

Không tương đương với

**\*p1 = \*p2**

- n **p1 = p2** có hiệu quả trỏ **p1** tới **y**, lệnh đó không thay đổi **x**
- n Lệnh **\*p1 = \*p2**; tương đương với **x = y**;

# Sử dụng typedef

- n Lỗi hay gặp khi sử dụng con trỏ. Phân biệt hai dòng sau:

```
int* P, Q; // P is a pointer and Q an int
```

```
int *P, *Q; // P and Q are both pointers
```

- n Một cách tránh lỗi là sử dụng lệnh typedef để đặt tên kiểu mới. Ví dụ:

```
typedef double distance; //distance is a new name for double
```

```
distance miles;
```

Giống như

```
double miles;
```

Có nghĩa rằng, thay vì viết

```
int *P, *Q;
```

Ta có thể viết

```
typedef int* IntPtr; // new name for pointers to ints
```

```
IntPtr P, Q; //P and Q are both pointers
```

# Cấp phát bộ nhớ tĩnh và động (Static and Dynamic Allocation Of Memory)

Đoạn trình

```
int X,Y; // X and Y are integers
```

```
int *P; // P is an integer pointer variable
```

Cấp phát bộ nhớ cho X, Y và P tại thời điểm biên dịch

Đó là **cấp phát tĩnh (static allocation)**

- n Bộ nhớ cũng có thể được cấp phát tại thời gian chạy. Đó gọi là **Cấp phát động (dynamic allocation)**. Ví dụ:

```
P = new int;
```

- .. Cấp phát một ô nhớ mới có thể chứa một số nguyên, và trỏ P tới ô nhớ đó

# Ví dụ

```
//Program to demonstrate pointers
//and dynamic variables
#include <iostream>
int main()
{
    int *p1, *p2;
    p1 = new int;
    *p1 = 10;
    p2 = p1;
    cout << "*p1 = " << *p1 << endl;
    cout << "*p2 = " << *p2 << endl << endl;

    *p2 = 30;
    cout << "*p1 = " << *p1 << endl;
    cout << "*p2 = " << *p2 << endl << endl;

    p1 = new int;
    *p1 = 40;
    cout << "*p1 = " << *p1 << endl;
    cout << "*p2 = " << *p2 << endl << endl;
}
```

```
*p1 = 10
*p2 = 10

*p1 = 30
*p2 = 30

*p1 = 40
*p2 = 30
```

# Cấp phát - thu hồi bộ nhớ động

- n **heap**: vùng bộ nhớ đặc biệt dành riêng cho các biến động. Để tạo một biến động mới, hệ thống cấp phát không gian từ heap. Nếu không còn bộ nhớ, **new** không thể cấp phát bộ nhớ thì nó trả về giá trị **NULL**
- n Trong lập trình thực thụ, ta nên luôn luôn kiểm tra lỗi này

```
int *p;
p = new int;
if (p == NULL) {
    cout << "Memory Allocation Error\n";
    exit;
}
```
- n Thực ra, NULL là giá trị 0, nhưng ta coi nó là một giá trị đặc biệt vì còn sử dụng cho trường hợp đặc biệt: con trỏ "rỗng".

# Cấp phát - thu hồi bộ nhớ động

- n Hệ thống chỉ có một lượng bộ nhớ giới hạn,
  - .. cần trả lại cho heap phần bộ nhớ động không còn sử dụng.

## n Lệnh

**delete P;**

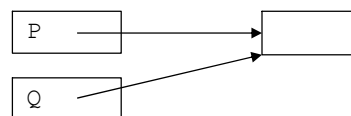
- .. trả lại vùng bộ nhớ trở bởi **P**, nhưng không sửa giá trị của **P**.
- .. Sau khi thực thi **delete P**, giá trị của **P** không xác định.

# Con trỏ lạc – Dangling Pointer

- n khi **delete P**, ta cần chú ý không xóa vùng bộ nhớ mà một con trỏ Q khác đang trỏ tới.

```
int *P;  
int *Q;  
P = new int;  
Q = P;
```

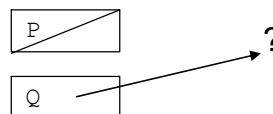
tạo



sau đó

```
delete P;  
P = Null;
```

làm Q bị lạc



# Rò rỉ bộ nhớ

- n Một vấn đề liên quan: *mất* mọi con trỏ đến một vùng bộ nhớ được cấp phát. Khi đó, vùng bộ nhớ đó bị mất dấu, không thể trả lại cho heap được.

```
int *P;  
int *Q;  
P = new int;  
Q = new int;
```

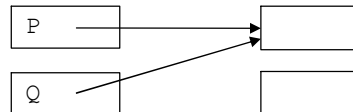
tạo



sau đó

```
Q = P;
```

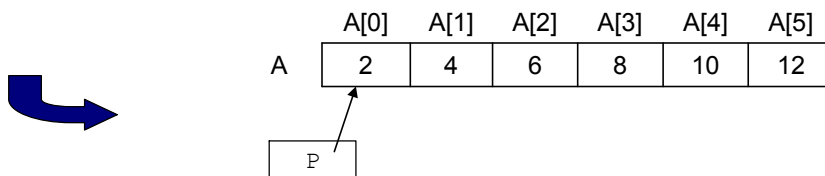
làm mất vùng  
nhớ đã từng  
được Q trỏ tới



# Mảng và con trỏ

Tên mảng được coi như một *con trỏ* tới phần tử đầu tiên của mảng.

```
int A[6] = {2,4,6,8,10,12}; // defines an array of inegers  
int *P;  
  
P = A; // P points to A
```

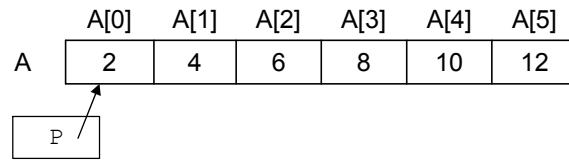


Do tên mảng và con trỏ là tương đương, ta có thể dùng P như tên mảng. Ví dụ:

**P[3] = 7; tương đương với A[3] = 7;**

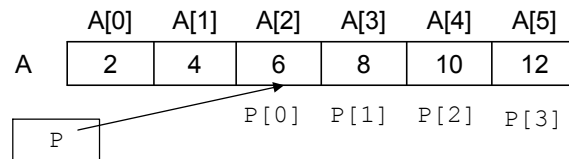
# Ví dụ

Bắt đầu



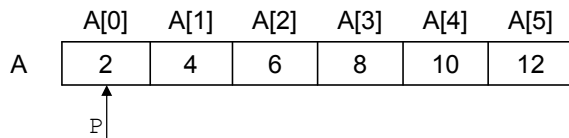
Thực hiện  $P = \&A[2]$

Bây giờ,  $P[0]$  là  $A[2]$ ,  
 $p[1]$  là  $A[3]$ ,...

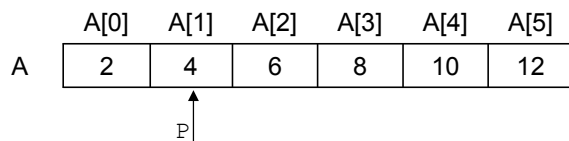


## Các phép tính trên con trỏ

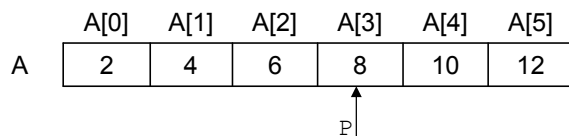
$P = A;$



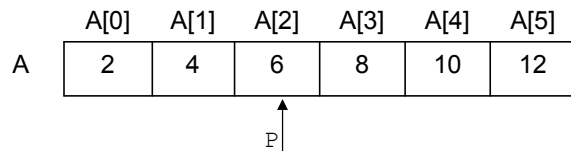
$P++;$



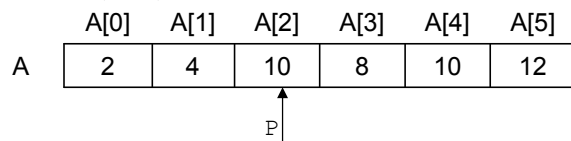
$P = P + 2;$



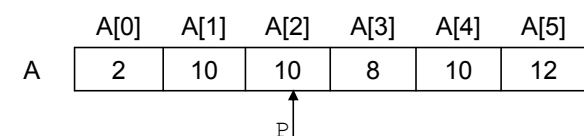
$P = A + 2;$



$*P = *(P+1) + 2;$



$*(P-1) = *(P+2)$





# Con trỏ tới bản ghi: bộ nhớ động

```
#ifndef IQ1_H
#define IQ1_H

#include <iostream>

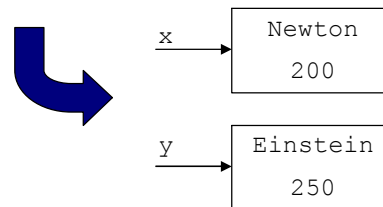
class IQ
{
private:
    char name[20];
    int score;
public:
    IQ (const char s, int k) {
        strcpy(name, s);
        score = k;
    }
    void smarter(int k) { score += k;}
    void print() const {
        cout << "(" << name << ", "
            << score << ")" << endl;
    }
}
#endif
```

```
#include <iostream>
#include "iq1.h"

int main()
{
    IQ *x = new IQ("Newton",200);
    IQ *y = new IQ("Einstein",250);

    x->print();
    y->print();

    return;
}
```



# Mảng cấp phát động

- n **new T[n]** cấp phát một mảng gồm n đối tượng kiểu T và trả về một con trỏ tới đầu mảng
- n **delete [] p** huỷ mảng mà p trỏ tới và trả vùng bộ nhớ đó cho heap. **p phải** trỏ tới đầu mảng động, Nếu không, kết quả của **delete** sẽ phụ thuộc vào trình biên dịch và loại dữ liệu đang sử dụng. Ta có thể nhận được lỗi runtime error hoặc kết quả sai.
- n Kích thước của mảng động không cần là hằng số mà có thể có giá trị được quyết định tại thời gian chạy

```
#include <iostream>
int main ()
{
    int size;
    cin << size;
    int* A = new int[size];        // dynamically allocate array

    A[0] = 0; A[1] = 1; A[2] = 2;
    cout << "A[1] = " << A[1] << endl;

    delete [] A;                // delete the array
}
```

# Hủy mảng động bất hợp lệ

P không trở tới đầu mảng A

Hủy không hợp lệ

Kết quả phụ thuộc trình biên dịch

```
#include <iostream>

int main ()
{
    int* A = new int[6];      // dynamically allocate array

    A[0] = 0; A[1] = 1; A[2] = 2;
    A[3] = 3; A[4] = 4; A[5] = 5;

    int *p = A + 2;
    cout << "A[1] = " << A[1] << endl;

    delete [] p;              // illegal!!!

    // results depend on particular compiler
    cout << "A[1] = " << A[1] << endl;
}
```

# Cấp phát động mảng đa chiều

- Cấp phát động mảng hai chiều  $(N+1)(M+1)$  gồm các đối tượng IQ:

```
IQ **a = new (IQ*) [N+1];
for (int i=0; i<N+1; i++)
    a[i] = new IQ[M+1];
```

