

# NGÔN NGỮ LẬP TRÌNH C/C++

Nguyễn Hải Châu  
Khoa Công nghệ thông tin  
Trường Đại học Công nghệ  
(Bài giảng tuần 5-6)

1

## Nội dung

- Con trỏ và số học địa chỉ
  - Con trỏ
  - Con trỏ và mảng
- Hàm và chương trình
  - Khai báo và sử dụng hàm
  - Các cách truyền đổi cho hàm

2

## Con trỏ và số học địa chỉ

3

## Khái niệm con trỏ

- Con trỏ là một biến chứa địa chỉ của một biến khác, hoặc địa chỉ của một hàm
- Nếu p là con trỏ chứa địa chỉ của biến x ta gọi p trỏ tới x và x được trỏ bởi p
- Để lấy địa chỉ của biến x, ta dùng phép toán **&**: **&x**
- Để lấy nội dung của con trỏ, ta dùng phép toán **\***: **\*p**

4

## Ví dụ về con trỏ, phép toán & và \*

```
int a=2; // a là một biến integer
int *p; // p là một con trỏ
```

```
p = &a; // p chứa địa chỉ của a
cout << p << endl; // Kết quả in ra là địa chỉ của a
cout << *p; // Kết quả in ra là 2
```

5

## Các phép toán với con trỏ

- Phép toán \* và &
- Phép toán gán: p = q; p và q là hai con trỏ
- Phép toán tăng giảm địa chỉ, tự tăng giảm
  - p+n, p-n
  - p++, p--, ++p, --p
- So sánh hai con trỏ: ==, >, >=, <, <=

6

## Cấp phát bộ nhớ cho con trỏ

- Để cấp phát bộ nhớ cho con trỏ, ta dùng chỉ thị **new**:  
p = new <kiểu> ; // cấp phát 1 phần tử  
p = new <kiểu>[n] ; // cấp phát n phần tử
- Ví dụ:  
int \*p, \*q;  
p = new int; // Cấp phát 1 phần tử  
q = new int[10]; // Cấp phát 10 phần tử

7

## Giải phóng bộ nhớ đã cấp phát

- Để cấp phát bộ nhớ cho con trỏ, ta dùng chỉ thị **delete**:  
delete p; // nếu p được cấp phát 1 phần tử  
delete[] p; // nếu p được cấp phát n>1 phần tử
- Ví dụ:  
int \*p, \*q;  
p = new int; // Cấp phát 1 phần tử  
q = new int[10]; // Cấp phát 10 phần tử  
delete p; // Giải phóng p  
delete[] q; // Giải phóng q

8

## Con trỏ và mảng một chiều

- Con trỏ trỏ đến mảng cũng tương tự trỏ đến các biến khác, tức gán địa chỉ của mảng (chính là tên mảng) cho con trỏ
- Địa chỉ của mảng là địa chỉ của thành phần đầu tiên (0) nên a+i sẽ là địa chỉ thành phần thứ i của mảng
- Giả sử có mảng int a[10]:
  - a[i] chính là \*(a+i)
  - a+i chính là &a[i]

9

## Con trỏ và mảng hai chiều

Ví dụ:

```
float a[2][3], *p;
```

a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]
a			a+1		

```
p = a;  
a[i][j] ~ *(p+3*i+j)
```

10

## Mảng con trỏ

- Khai báo:  
○ <kiểu> \* <tên mảng con trỏ>[<số lượng>];
- Ví dụ:  
int \*a[10]; // Mảng 10 con trỏ số nguyên
- Ví dụ: khai báo tham số của hàm main:  
main(argc, argv)  
int argc;  
char \*argv[];

11

Hàm

12

## Khái niệm về hàm

- Hàm là một chương trình con
- Hàm có thể nhận hoặc không nhận đối số
- Hàm có thể trả lại kết quả hoặc không
- Một chương trình C chứa ít nhất một hàm (main) và có thể có nhiều hàm khác
- Hàm giúp cho việc phân đoạn chương trình thành những môđun độc lập

13

## Đặc trưng của hàm

- Nằm trong hoặc ngoài văn bản có chương trình gọi đến hàm. Trong một văn bản có thể chứa nhiều hàm,
- Được gọi từ chương trình chính (main), từ hàm khác hoặc từ chính nó (đệ quy),
- Không lồng nhau.
- Có 3 cách truyền giá trị: Truyền theo tham trị, tham biến và tham trở.

14

## Khai báo hàm

- Khai báo hàm:
  - <kiểu giá trị trả lại> <tên hàm>(d/s kiểu đối);
- Ví dụ:
  - int myfunction(int, long);
  - int rand100();
  - void showtext(char \*);
  - void nothing();

15

## Định nghĩa hàm

```
<kiểu giá trị trả về> <tên hàm>(danh sách tham đối hình thức)
{
    khai báo cục bộ của hàm ; // chỉ dùng cho hàm này
    dãy lệnh của hàm ;
    return (biểu thức trả về); // có thể nằm đâu đó trong dãy
    lệnh.
}
```

16

## Ví dụ

```
double luythua(float x, int n)
{
    int i ; // biến chỉ số
    double kq = 1; // để lưu kết quả
    for (i=1; i<=n; i++) kq *= x;
    return kq;
}
```

17

## Lời gọi hàm

<tên hàm>(danh sách tham đối thực sự);

Ví dụ:

Viết và thực hiện một chương trình đơn giản có sử dụng lời gọi hàm

18

## Hàm với đối ngầm định

- Khai báo:  
<kiểu hàm> <tên hàm>(d1, ..., dn,  
dnd1=gt1, ..., dndm=gtm);
- Các đối ngầm định phải được khai báo liên tục và nằm ở cuối danh sách đối
- Ví dụ:
  - int function(int, char, int=0, float=1.0);
  - int=0 và float=1.0 chỉ ra hai đối với giá trị ngầm định

19

## Khai báo hàm trùng tên (Overlay)

```
int max(int a, int b)      int max(double a, double b)
{                          {
    return (a > b) ? a : b;  return (a > b) ? a : b;
}
```

20

## Biến tham chiếu

Biến tham chiếu

```
int i;
int &j=i; // j là một cách tham chiếu khác
         // của biến i
j = 5;   // Sau lệnh gán này i cũng có giá trị 5
```

**Biến tham chiếu phải được khởi tạo khi khai báo**

21

## Các cách truyền đối cho hàm

- Truyền theo tham trị
- Truyền theo tham chiếu
- Truyền theo con trỏ

22

## Truyền theo tham trị

```
void swap1(int x, int y)
{
    int t; t = x; x = y; y = t;
}
main()
{
    int x=5, y=6;
    cout << "x = " << x << " y = " << y << endl;
    swap1(x, y);
    cout << "x = " << x << " y = " << y << endl;
}
```

23

## Truyền theo tham trỏ

```
void swap2(int *x, int *y)
{
    int t; t = *x; *x = *y; *y = t;
}
main()
{
    int x=5, y=6;
    cout << "x = " << x << " y = " << y << endl;
    swap2(&x, &y);
    cout << "x = " << x << " y = " << y << endl;
}
```

24

## Truyền theo tham chiếu

```
void swap3(int &x, int &y)
{
    int t; t = x; x = y; y = t;
}
main()
{
    int x=5, y=6;
    cout << "x = " << x << " y = " << y << endl;
    swap3(x, y);
    cout << "x = " << x << " y = " << y << endl;
}
```

25

	Tham trị	Tham chiếu	Tham trỏ
Khai báo đối	void swap(int x, int y)	void swap(int &x, int &y)	void swap(int *x, int *y)
Câu lệnh	t = x; x = y; y = t;	t = x; x = y; y = t;	t = *x; *x = *y; *y = t;
Lời gọi	swap(a, b);	swap(a, b);	swap(&a, &b);
Tác dụng	a, b không thay đổi	a, b có thay đổi	a, b có thay đổi

26

## Các vấn đề cần nhớ

- Con trỏ: Cách khai báo, sử dụng, cấp phát và giải phóng bộ nhớ
- Mối liên quan giữa con trỏ và mảng
- Khai báo, xây dựng và sử dụng hàm
- Phân biệt các cách truyền đối khác nhau cho hàm
- Đối ngầm định, hàm trùng tên

27

## Bài tập

- Các bài tập từ số 1 đến số 42 của chương 4 (Từ trang 140-144)

28