

# NGÔN NGỮ LẬP TRÌNH C/C++

Nguyễn Hải Châu  
Khoa Công nghệ thông tin  
Trường Đại học Công nghệ  
(Bài giảng tuần 9)

1

## Nội dung

- Các phương pháp luận lập trình
  - Lập trình cấu trúc
  - Lập trình hướng đối tượng
- Cơ sở lập trình hướng đối tượng trong C++
  - Đối tượng
  - Lớp

2

## Lập trình cấu trúc

- Lập trình cấu trúc: tổ chức chương trình thành các chương trình con (hàm hoặc thủ tục)
- Hàm là một đơn vị chương trình độc lập dùng để thực hiện một công việc nào đó
- Trao đổi dữ liệu giữa các hàm thực hiện thông qua các đối và các biến toàn cục

3

## Lập trình cấu trúc (tiếp)

- Một chương trình "cấu trúc" gồm các cấu trúc dữ liệu (như biến, mảng, bản ghi) và các hàm, thủ tục.
- Nhiệm vụ chính của việc thiết kế chương trình cấu trúc là tổ chức chương trình thành các hàm, thủ tục.

4

## Lập trình hướng đối tượng

- Lập trình hướng đối tượng có thể được xem là lập trình có cấu trúc kết hợp *trừu tượng hóa dữ liệu*
- Việc thiết kế chương trình chú trọng vào dữ liệu
- Dữ liệu và các thao tác trên dữ liệu được gắn kết chặt chẽ với nhau (khác với lập trình cấu trúc)

5

## Lập trình hướng đối tượng (tiếp)

- Lập trình hướng đối tượng được xây dựng dựa trên đặc trưng chính là khái niệm *đóng gói*
- Đóng gói là khái niệm trung tâm của lập trình hướng đối tượng: dữ liệu và các thao tác xử lý được qui định trước và "đóng" thành một "gói" thống nhất, riêng biệt với các dữ liệu khác tạo thành kiểu dữ liệu với tên gọi là các *lớp (class)*

6

## Lớp và đối tượng

- Lớp (class) là khái niệm quan trọng nhất của lập trình hướng đối tượng
- Một lớp đơn bao gồm các hàm và dữ liệu có liên quan
- Các hàm: hàm thành phần/phương thức (member function/method)
- Các hàm qui định các thao tác được phép thực hiện trên dữ liệu của lớp

7

## Khai báo lớp

```
class tên_lớp
{
    // Khai báo các thành phần dữ liệu (thuộc tính)
    // Khai báo các phương thức (hàm)
};
```

8

## Khai báo lớp (tiếp)

- Thuộc tính của lớp có thể là các biến, mảng, con trỏ có kiểu chuẩn (int, float, char, char\*, long,...) hoặc kiểu ngoài chuẩn đã định nghĩa trước (cấu trúc, hợp, lớp,...).
- Thuộc tính của lớp không thể có kiểu của chính lớp đó, nhưng có thể là con trỏ của lớp này

9

## Ví dụ về khai báo lớp

```
class daydiem
{
    int n;
    float *x,*y;
    public:
    float do_dai(int i, int j)
    {
        return sqrt(pow(x[i]-x[j],2)+pow(y[i]-y[j],2));
    }
    void nhapsl(void);
};
```

10

## Ví dụ về khai báo lớp (tiếp)

```
void daydiem::nhapsl(void)
{
    int i;
    printf("\n So diem N= ");
    scanf("%d",&n);
    x = (float*)malloc((n+1)*sizeof(float));
    y = (float*)malloc((n+1)*sizeof(float));
    for (i=1; i<=n; ++i)
    {
        printf("\n Nhap toa do x, y cua diem thu %d : ",i);
        scanf("%f%f",&x[i],&y[i]);
    }
}
```

11

## Khai báo lớp có thành phần tự trỏ

```
class A
{
    A x; //Không cho phép, vì x có kiểu lớp A
    A* p; //Cho phép, vì p là con trỏ kiểu lớp A
};
```

12

## Khai báo các thành phần của lớp

- Các từ khóa private và public:
  - Các thành phần khai báo private chỉ có thể được truy cập từ bên trong lớp
  - Các thành phần khai báo public có thể được truy cập từ trong hoặc ngoài
- Sử dụng private có tác dụng che giấu thông tin của mỗi lớp

13

## Khai báo thành phần dữ liệu

- Được khai báo như khai báo các thành phần trong kiểu cấu trúc hay hợp
- Các thành phần này thường được khai báo là private để bảo đảm an toàn dữ liệu của lớp, không cho phép các hàm bên ngoài xâm nhập vào các dữ liệu này.

14

## Khai báo hàm (phương thức)

- Thường khai báo là public để chúng có thể được gọi tới từ các bên ngoài lớp
- Các phương thức có thể được khai báo và định nghĩa bên trong lớp hoặc chỉ khai báo bên trong còn định nghĩa cụ thể của phương thức có thể được viết bên ngoài
- Thông thường, các phương thức ngắn được định nghĩa bên trong lớp, các phương thức dài viết bên ngoài lớp

15

## Ví dụ: Khai báo lớp

```
class DIEM
{
    private:
        int x, y, m ;
    public:
        void nhapsl() ;
        void hien() ;
        void an() { putpixel(x, y, getbkcolor());}
};
```

16

## Định nghĩa phương thức nhapsl()

```
void DIEM::nhapsl()
{
    cout << "\n Nhập hoành do (cot) va tung do (hang) cua diem: ";
    cin >> x >> y ;
    cout << "\n Nhập ma mau cua diem: ";
    cin >> m ;
}
```

17

## Định nghĩa phương thức hien()

```
void DIEM::hien()
{
    int mau_ht ;
    mau_ht = getcolor();
    putpixel(x, y, m);
    setcolor(mau_ht);
}
```

18

## Biến, mảng, con trỏ đối tượng

- Một lớp sau khi định nghĩa có thể xem như một kiểu đối tượng và có thể dùng để khai báo các biến, mảng đối tượng
- Cách khai báo giống khai báo biến thông thường:  
**Tên\_lớp danh sách biến, mảng, con trỏ;**
- Ví dụ:
  - DIEM d1, d2, d3 ;
  - DIEM d[20] ;

19

## Truy cập thuộc tính, phương thức

- Truy cập thuộc tính:
  - tên\_lớp.tên\_thuộc\_tính
  - tên\_con\_trỏ\_lớp->tên\_thuộc\_tính
- Truy cập thuộc tính:
  - tên\_lớp.tên\_phương\_thức[danh\_sách\_đối]
  - tên\_con\_trỏ\_lớp.tên\_phương\_thức[danh\_sách\_đối]

20

## Bài tập

- Thiết kế lớp vector:
  - Thể hiện cấu trúc dữ liệu của vector 3 chiều
  - Viết các phương thức thực hiện các phép toán trên vector: Cộng, trừ hai vector, nhân vector với một số, chuẩn hóa vector, tích vô hướng của hai vector 3 chiều

21