Mở đầu

Lập trình hướng đối tượng

Giới thiệu môn học

- Phần C++ đã học chỉ đủ để viết những chương trình C++ nhỏ.
- Sinh viên gặp nhiều khó khăn khi viết một chương trình lớn hoặc làm việc trong một nhóm cùng các lập trình viên khác? Làm thế nào để đảm bảo tính thống nhất trong chương trình lớn hoặc giữa các lập trình viên khác nhau?
- Trong môn học này, sinh viên sẽ học những kiến thức cơ bản về lập trình hướng đối tượng (Object-Oriented Programming - OOP). Mục đích là để học cách thiết kế và viết những dự án phần mềm lớn.

Lịch sử ngắn gọn

Các ngôn ngữ lập trình thời kỳ đầu như Basic, Fortran... không có cấu trúc và cho phép viết những đoạn mã rối rắm (spaghetti code). Lập trình viên sử dụng các lệnh goto" và "gosub" để nhảy đến mọi nơi trong chương trình.

```
10      k=1
20      gosub 100
30      if y > 120 goto 60
40      k = k + 1
50      goto 20
60      print k, y
70      stop
100     y = 3*k*k + 7*k - 3
110     return
```

Lệnh nhảy đến vị trí bất kỳ trong chương trình

Đoạn trình trên khó theo dõi, khó hiểu, dễ gây lỗi, khó sửa đổi.

Lịch sử ngắn gọn...

Kiểu lập trình rối rắm trên dẫn tới phong cách lập trình mới: lập trình cấu trúc, với các ngôn ngữ Algol, Pascal, C...

```
int func(int j)
{
   return (3*j*j + 7*j-3;
}

int main()
{
   int k = 1
   while (func(k) < 120)
      k++;
   printf("%d\t%d\n", k, func(k));
   return(0);
}
```

Đặc điểm của lập trình cấu trúc hay lập trình thủ tục (Procedural Programming - PP) là:

- Sử dụng các cấu trúc vòng lặp: for, while, repeat, do-while
- Chương trình là một chuỗi các hàm/ thủ tục
- Mã chương trình tập trung thể hiện thuật toán: làm như thế nào.

Hạn chế của lập trình thủ tục

- dữ liệu và phần xử lý tách biệt
- dữ liệu thụ động, xử lý chủ động
- không đảm bảo được tính nhất quán và các ràng buộc của dữ liệu
  - khó cấm mã ứng dụng sửa dữ liệu của thư viện
- khó bảo trì code
  - phần xử lý có thể nằm rải rác và phải hiểu rõ cấu trúc dữ liệu

```
struct Date
{
   int day;
   int month;
   int year;
};

void setDate(Date& date,
             int newDay, int newMonth, int newYear)
{
   date.day = newDay;
   ...
}
...
```

Chuyện gì xảy ra nếu các đối số **`newDay, newMonth, newYear`** tạo thành ngày tháng năm không hợp lệ?

Lập trình hướng đối tượng

- Lập trình hướng đối tượng cho phép khắc phục các hạn chế nói trên

```
class Date
{
public:
   void setDate(int newDay, int newMonth, int newYear);
   int getDay() { return day; }
   ...
private:
   int day;
   int month;
   int year;
};

void Date::setDate(int newDay, int newMonth, int newYear)
{
   //check validity of newDay, newMonth, newYear
   ...
   //set new values
   ...
}
```

Lập trình hướng đối tượng

Chú ý: Lập trình hướng đối tượng không tự dưng cho ta thiết kế chương trình tốt.

- Ví dụ: hai đoạn trình dưới đây không có gì khác nhau. Thậm chí đoạn chương trình hướng đối tượng còn tồi hơn.

```
struct Date
{
   int day;
   int month;
   int year;
};
```

```
class Date
{
public:
   int getDay { return day;}
   ...
   int setDay(int newDay) { day = newDay; }
   ...
private:
   int day;
   int month;
   int year;
};
```

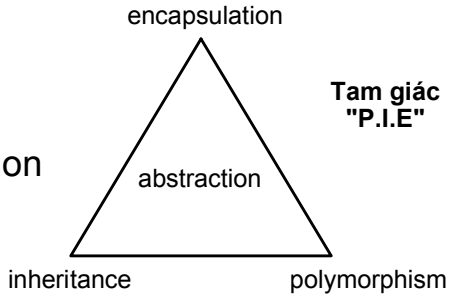

Lịch sử OOP

- Các ngôn ngữ lập trình hướng đối tượng không mới
  - Simula (1967) là ngôn ngữ đầu tiên, có lớp, thừa kế, liên kết động (hay còn gọi là hàm ảo)
- Nhưng các ngôn ngữ hướng đối tượng chậm hơn các ngôn ngữ thời kỳ đầu
  - nên chúng chỉ được dùng rộng rãi khi máy tính bắt đầu chạy nhanh (khoảng thời gian chiếc máy Pentium đầu tiên ra đời)
  - Lưu ý rằng biên dịch các chương trình hướng đối tượng cũng chậm

Hướng đối tượng là gì?

- Một số hệ thống “hướng đối tượng” thời kỳ đầu không có các lớp
  - chỉ có các “đối tượng” và các “thông điệp” (v.d. Hypertalk)
- Hiện giờ, đã có sự thống nhất rằng hướng đối tượng là:
  - lớp - class
  - thừa kế - inheritance và liên kết động - dynamic binding
- Một số đặc tính của lập trình hướng đối tượng có thể được thực hiện bằng C hoặc các ngôn ngữ lập trình thủ tục khác.
- Điểm khác biệt sự hỗ trợ và ép buộc ba khái niệm trên được cài hẳn vào trong ngôn ngữ.
- Mức độ hướng đối tượng của các ngôn ngữ không giống nhau
  - Eiffel (tuyệt đối), Java (rất cao), C++ (nửa nọ nửa kia)

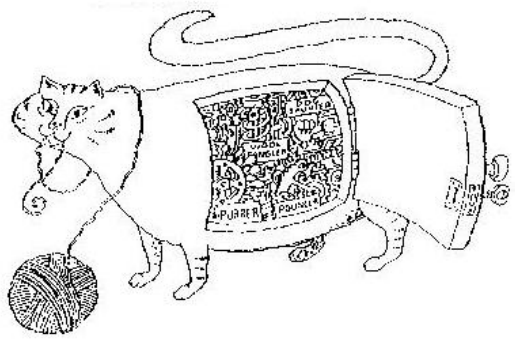
Các đặc điểm quan trọng của OO



Tam giác "P.I.E"

- Các lớp đối tượng - Classes
  - Khả năng lưu trạng thái - State retention
  - Định danh đối tượng - Object identity
  - Các thông điệp - Messages
- Đóng gói – Encapsulation
  - Che dấu thông tin - Information/implementation hiding
- Thừa kế - Inheritance
- Đa hình - Polymorphism
- Lập trình tổng quát - Genericity

Đóng gói
Che dấu thông tin



- Đóng gói: Nhóm những gì có liên quan với nhau vào làm một, để sau này có thể dùng một cái tên để gọi đến
  - Các hàm/ thủ tục đóng gói các câu lệnh
  - Các đối tượng đóng gói dữ liệu của chúng và các thủ tục có liên quan
- Che dấu thông tin: đóng gói để che một số thông tin và chi tiết cài đặt nội bộ để bên ngoài không nhìn thấy
  - mục tiêu là để khách hàng của ta (thường là các lập trình viên khác) coi các đối tượng của ta là các hộp đen

Đối tượng

- ***Lưu giữ trạng thái***: mỗi đối tượng có trạng thái (dữ liệu của nó) và các thao tác
  - có thể nói đối tượng có một dạng "ký ức" về quá khứ
  - các thao tác của đối tượng có thể sửa trạng thái của đối tượng đó.
- ***Định danh***: Mỗi đối tượng bất kể đang ở trạng thái nào đều có định danh và được đối xử như một thực thể riêng biệt.
  - mỗi đối tượng có một *handle* (trong C++ là địa chỉ)
  - hai đối tượng có thể có giá trị giống nhau nhưng *handle* khác nhau
  - ngôn ngữ Lisp phân biệt hai phép so sánh **`eq`** và **`equal`**

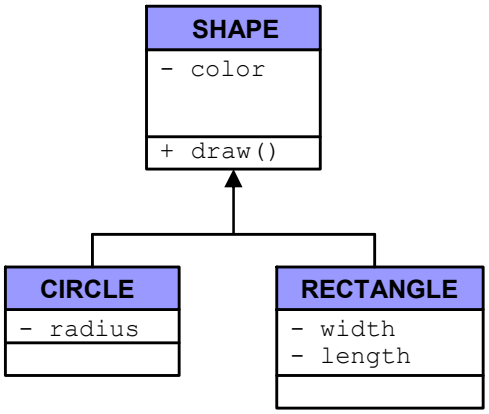
Đối tượng

- ***Thông điệp***: là phương tiện để một đối tượng A chuyển tới đối tượng B yêu cầu B thực hiện một trong số các thao tác của B.
- một thông điệp gồm 3 phần:
  - handle của đối tượng đích (đối tượng chủ)
  - tên thao tác cần thực hiện
  - các thông tin cần thiết khác (các đối số)
- thực ra là một lời gọi hàm với một đối số ẩn là đối tượng chủ
- tuy nhiên, khái niệm thông điệp có ý nghĩa lớn đối với OOP
  - dữ liệu trở nên chủ động
  - đối lập với quan điểm cũ về lập trình rằng
    - mọi hành động được điều khiển tập trung
    - dữ liệu luôn luôn bị động, các thủ tục thao tác dữ liệu

Lớp đối tượng - class

- Lớp: là khuôn mẫu để tạo các đối tượng (tạo các thể hiện). Mỗi đối tượng có cấu trúc và hành vi giống như lớp đối tượng mà nó được tạo từ đó
  - VD. Lớp VánCờ, các ván cờ cụ thể là các đối tượng VánCờ
- Lớp là cái ta thiết kế và lập trình
- Đối tượng là cái ta tạo (từ một lớp) tại thời gian chạy

Thừa kế



- là cơ chế cho phép một lớp **D** có được các thuộc tính và thao tác của lớp **C**, như thể các thuộc tính và thao tác đó đã được định nghĩa lại lớp **D**.
- cho phép các phần mềm sử dụng quan hệ "là"
- giúp ta thiết kế các dịch vụ tổng quát rồi chuyên môn hóa chúng

Đa hình

- Đa hình hàm - Functional polymorphism
  - cơ chế cho phép một tên thao tác hoặc thuộc tính có thể được định nghĩa tại nhiều lớp và có thể có nhiều cài đặt khác nhau tại mỗi lớp trong các lớp đó
    - v.d. lớp Date cài 2 phương thức setDate(),một nhận tham số là một đối tượng Date, phương thức kia nhận 3 tham số day, month, year.
- Đa hình đối tượng - Object polymorphism
  - các đối tượng thuộc các lớp khác nhau có khả năng hiểu cùng một thông điệp theo các cách khác nhau
    - vd. khi nhận được cùng một thông điệp draw(), các đối tượng Rectangle và Triangle hiểu và thực hiện các thao tác khác nhau

Lập trình tổng quát – genericity

- khả năng xây dựng một lớp **C** sao cho một hoặc nhiều lớp được sử dụng bên trong C sẽ được cung cấp tại thời gian chạy
  - vd. kiểu ngăn xếp tổng quát, có thể dùng để chứa các đối tượng Date, có thể dùng cho các string…
- Thực tế với C++:
  - cài đặt bằng khuôn mẫu – template hay các kiểu có tham số
  - Lựa chọn khuôn mẫu được quyết định tại thời điểm biên dịch của chương trình khách hàng - user
    - không phải tại thời điểm biên dịch của chương trình của người thiết kế - thư viện