

# BÀI THỰC HÀNH TUẦN 6

## Nội dung:

- Tính đa hình trong C++

## Qui định nộp bài tập:

- Mỗi bài tập tương ứng với 1 project, tên Project là BaiXX, với XX là thứ tự của bài tập (Ví dụ bài tập 1 tên Project tương ứng là Bai02).
- Tất cả các bài tập được đặt bên trong một thư mục, tên thư mục theo qui định như sau: **BTH6\_HoVaTen\_MSSV**. Ví dụ Sinh viên Nguyễn Sơn Trà có MSSV là 19521269 thì đặt tên thư mục như sau: **BTH6\_NguyenSonTra\_19521269**
- Sau đó nén thư mục trên thành tập tin **.zip hoặc .rar** (tên file nén cũng theo qui định như tên thư mục). Ví dụ **BTH6\_NguyenSonTra\_19521269.rar**
- Lưu ý xóa thư mục được phát sinh sau khi biên dịch (thư mục Debug, .vs) của mỗi project
- Hình thức nộp bài: Nộp trên website môn học theo thời gian qui định.
- **Những bài nộp không đúng qui định như trên sẽ không được chấm điểm (0 điểm)**
- **Tất cả các bài làm có tính chất sao chép (copy) sẽ nhận 0 điểm**

## 1. Tính chất đa hình trong lập trình hướng đối tượng

### a. Liên kết tĩnh, liên kết động và phương thức ảo

- Trong quan hệ kế thừa, bên cạnh việc tạo trực tiếp một đối tượng cụ thể, ta có thể sử dụng biến con trỏ để lưu trữ đối tượng cần tạo. Cơ chế trong quan hệ kế thừa cho phép tạo một biến con trỏ có kiểu dữ liệu là lớp cha, trỏ đến đối tượng kiểu lớp con
  - Xem ví dụ sau, ta có quan hệ giữa heo, mèo và động vật là quan hệ kế thừa, trong đó con heo, con mèo kế thừa từ lớp động vật.

```
class DongVat
{
    //Các thuộc tính...
public:
    void TiengKeu()
    {
        cout << "Chua biet dong vat gi!!" << endl;
    }
};

class ConHeo : public DongVat
{
    //Các thuộc tính riêng của con heo
public:
    void TiengKeu()
    {
        cout << "Ut Ut!!" << endl;
    }
};

class ConMeo : public DongVat
{
    //Các thuộc tính riêng của con mèo
public:
    void TiengKeu()
    {
        cout << "Meo meo!!" << endl;
    }
};
```

```

void main()
{
    ConMeo conmeo1;
    DongVat* conmeo2;
    //Con trở có kiểu của lớp cha ĐƯỢC
    //tham chiếu đến đối tượng lớp con
    conmeo2 = &conmeo1;
    conmeo2->TiengKeu();

    //Con trở có kiểu của lớp cha ĐƯỢC
    //tham chiếu đến đối tượng lớp con
    DongVat* conheo1 = new ConHeo();
    conheo1->TiengKeu();

    delete conheo1;
}

```

```

Chua biet dong vat gi!!
Chua biet dong vat gi!!
Press any key to continue . . .

```

- Dựa trên tính chất cho phép trở đến những lớp con trong quan hệ kế thừa, ta có thể sử dụng nạp chồng hàm (function overloading) để cài đặt mã nguồn thao tác trên đối tượng của lớp cha vẫn có thể chạy được cho đối tượng của lớp con, tuy nhiên cần kết hợp với tính đa hình.
  - Lưu ý: chỉ cho phép biến con trở có kiểu dữ liệu lớp cha trở đến đối tượng lớp con chứ không được phép làm ngược lại : **biến con trở có kiểu dữ liệu lớp con không được phép trở đến đối tượng lớp cha**
- Trong ví dụ trên, ta có thể thấy khi biến con trở mang kiểu dữ liệu của lớp cha (lớp Động Vật) trở đến đối tượng lớp con (con mèo, con heo), khi gọi phương thức nạp chồng (tiếng kêu) thì đều gọi lại phương thức này ở lớp cha !!! Đây chính là **liên kết tĩnh**.
- Để có thể gọi được đúng phương thức của đối tượng mà con trở đang trở đến, cần phải cài đặt **cơ chế liên kết động cho phương thức ở lớp cha**, bằng cách thêm từ khóa **virtual**
  - Phương thức nào được cài đặt liên kết động bằng cách thêm từ khóa virtual được gọi là **phương thức ảo hay hàm ảo**
- Phương thức ảo hay hàm ảo mang **tính chất đa hình**:
  - Sử dụng con trở có kiểu thuộc lớp cha: **Trở đến đối tượng thuộc lớp con nào thì gọi hàm hay phương thức của lớp con đó tương ứng**
- Để cài đặt một phương thức hay thuộc tính mang tính đa hình:
  - Cần phải có quan hệ kế thừa, cần phải sử dụng biến con trở với kiểu ở lớp cha, trở đến đối tượng lớp con
  - Xác định phương thức nào cần đa hình, thêm từ khóa virtual cho phương thức đó ở lớp cha
  - Lớp con cài đặt nạp chồng phương thức đó (không thêm virtual ở phương thức của lớp con, có thể thêm nếu kế thừa nhiều tầng !!!)

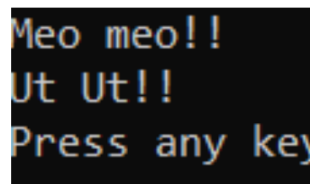
```

class DongVat
{
    //Các thuộc tính...
public:
    virtual void TiengKeu() //Phương thức ảo
    {
        cout << "Chua biet dong vat gi!!" << endl;
    }
};

void main()
{
    ConMeo conmeo1;
    DongVat* conmeo2;
    //Con trỏ có kiểu của lớp cha ĐƯỢC
    //tham chiếu đến đối tượng lớp con
    conmeo2 = &conmeo1;
    conmeo2->TiengKeu(); //Đa hình

    //Con trỏ có kiểu của lớp cha ĐƯỢC
    //tham chiếu đến đối tượng lớp con
    DongVat* conheo1 = new ConHeo();
    conheo1->TiengKeu(); //Đa hình
    delete conheo1;
}

```



```

Meo meo!!
Ut Ut!!
Press any key

```

- Cài đặt tính kế thừa kết hợp với đa hình giúp chương trình viết ngắn hơn, mang tính tổng quát đồng thời có khả năng mở rộng, tùy biến linh hoạt
- Xem xét ví dụ về khả năng mở rộng và viết mã nguồn ngắn hơn :
  - Giả sử ta cần cài đặt hàm để phát ra tiếng kêu của bất kỳ động vật nào truyền vào
    - Nếu không cài đặt đa hình : cần cài thêm xử lý riêng cho mỗi lớp con thêm mới vào do chưa biết cụ thể động vật truyền vào là động vật gì.
    - Nếu có cài đặt đa hình : không cần cài xử lý riêng cho mỗi lớp con, lớp con khi thêm mới đã tự cài đặt sẵn nạp chồng phương thức đa hình.
  - Minh họa mã nguồn cho cả 2 cách cài đặt :

```

//KHÔNG sử dụng tính đa hình
void PhatRaTiengKeu(DongVat dv, string loaidv)
{
    if (loaidv == "ConHeo")
    {
        //Xử lý cho trường hợp con heo
    }
    else if (loaidv == "ConMeo")
    {
        //Xử lý cho trường hợp con mèo
    }
    //Nếu có thêm con khác:
    //Phải cài lớp con, phải viết thêm xử lý cho mỗi lớp con
    //else if (loaidv == "ConVit")
    //.... con gà, con chó, ....
}

//CÓ sử dụng tính đa hình (cài đặt virtual)
void PhatRaTiengKeu(DongVat *dv)
{
    //Tính đa hình tổng quát
    //Thêm bất kỳ lớp con nào cũng không cần viết xử lý thêm
    dv->TiengKeu();
}

```

## b. Phương thức thuần ảo, hàm thuần ảo (pure virtual) và lớp trừu tượng (abstract class)

- Khi cài đặt kế thừa và đa hình, thường có xu hướng đưa những cái chung lên ở lớp cha và cài đặt xử lý linh hoạt ở lớp con. Do đó, lớp cha thường sẽ không có ý nghĩa ứng dụng trong thực tế, vậy không nên tạo ra đối tượng của lớp cha.
  - Lớp con heo, con mèo đều kế thừa từ lớp động vật. Ta không thể tạo đối tượng cho lớp động vật do lớp này mang ý nghĩa trừu tượng (chưa xác định được động vật này tên gọi là gì, có tiếng kêu như thế nào ?) mà phải tạo đối tượng cụ thể là con heo hay con mèo.
  - Cách xử lý tốt hơn: ở lớp cha phương thức ảo chỉ có khai báo, không có phần cài đặt
- Các phương thức ảo chỉ có khai báo và không có phần cài đặt được gọi là phương thức thuần ảo (hay hàm thuần ảo – pure virtual method), cài đặt bằng việc gán virtual = 0
- Lớp nào có chứa phương thức thuần ảo được gọi là lớp trừu tượng.
- **Các lớp con kế thừa từ lớp trừu tượng bắt buộc phải cài lại toàn bộ các phương thức thuần ảo có khai báo ở lớp cha**

```

class DongVat //Lớp cha chính là lớp trừu tượng do có 1 phương
thức thuần ảo
{
    //Các thuộc tính...
public:
    //Khai báo phương thức thuần ảo
    virtual void TiengKeu() = 0;
};

class ConHeo : public DongVat
{
public:
    //Lớp con kế thừa bắt buộc phải cài lại phương thức thuần ảo
    void TiengKeu()
    {
        cout << "Ut Ut!!" << endl;
    }
};

class ConMeo : public DongVat
{
    //Các thuộc tính riêng của con mèo
public:
    //Lớp con kế thừa bắt buộc phải cài lại phương thức thuần ảo
    void TiengKeu()
    {
        cout << "Meo meo!!" << endl;
    }
}

```

### c. Phương thức hủy ảo (hàm hủy ảo – virtual destructor)

- Do sử dụng tính chất đa hình, cần phải sử dụng con trỏ có kiểu thuộc lớp cha, trỏ đến đối tượng thuộc lớp con nào thì gọi hàm hay phương thức của lớp con đó tương ứng.
- Việc khai báo con trỏ ở lớp cha trỏ đến đối tượng ở lớp con, khi được khai báo, phương thức tạo lập ở lớp cha sẽ được gọi trước, sau đó đến phương thức tạo lập ở lớp con.
- Tuy nhiên, phương thức hủy chỉ được gọi theo đối tượng của lớp cha, do biến con trỏ ban đầu tạo lập là biến con trỏ trỏ đến kiểu dữ liệu của lớp cha
  - **Để gọi được phương thức hủy của lớp con kế thừa, cần cài đặt phương thức hủy ở lớp cha là phương thức ảo, được gọi là phương thức hủy ảo.**
- Trong trường hợp lớp con có biến kiểu con trỏ, nếu không cài đặt phương thức hủy ảo ở lớp cha, lớp con không được hủy, dẫn đến việc không gọi delete con trỏ được, chương trình sẽ bị **Memory Leak**.
- Ví dụ minh họa sau:

```

class DongVat
{
    //Các thuộc tính...
public:
    //Cài đặt phương thức hủy ảo để được gọi ở lớp con
    virtual ~DongVat()
    {
        cout << "Dong vat bi huy!!" << endl;
    }
};
class ConHeo : public DongVat
{
    int *cannang;
public:
    ~ConHeo()
    {
        delete cannang;
        cout << "Con heo bi huy!!" << endl;
    }
};

void main()
{
    //Constructor của lớp cha được gọi trước
    //Constructor của lớp con được gọi sau
    DongVat* conheo1 = new ConHeo();
    conheo1->TiengKeu(); //Đã hình

    //Chỉ phương thức hủy của lớp cha được gọi
    //Nguyên nhân: biến con trở khai báo ban đầu là kiểu lớp cha
    //Cần cài đặt hủy ảo ở lớp cha để gọi hủy ở lớp con trước,
    rồi đến hủy lớp cha
    delete conheo1;
}

```

## 2. Bài tập thực hành

1. Một công ty ABC muốn quản lý thông tin của các cán bộ tại cơ quan. Mỗi cán bộ trong cơ quan cần quản lý các thông tin chung như sau: mã số cán bộ, họ và tên, ngày sinh và lương

Cơ quan có hai loại cán bộ là: cán bộ thuộc biên chế nhà nước và cán bộ hợp đồng. Tùy theo cán bộ biên chế hay cán bộ hợp đồng mà cơ quan có cách tính lương khác nhau.

Đối với cán bộ thuộc biên chế nhà nước ngoài các thông tin chung cơ quan cần quản lý thêm các thông tin như sau: lương cơ bản, hệ số lương, hệ số phụ cấp

Đối với cán bộ hợp đồng ngoài các thông tin chung cơ quan cần quản lý thêm các thông tin như sau: tiền công, số ngày công và hệ số vượt giờ.

Yêu cầu: Viết chương trình cài đặt các lớp và các phương thức cần thiết để thực hiện các yêu cầu sau đây:

- a. Cho phép người dùng nhập danh sách cán bộ từ bàn phím (cán bộ biên chế, cán bộ hợp đồng)
- b. Xuất danh sách cán bộ ra màn hình (cán bộ biên chế, cán bộ hợp đồng)
- c. Tính lương cho những cán bộ trong cơ quan theo quy định sau:

- Đối với cán bộ thuộc biên chế nhà nước:

$$\text{Lương} = \text{Lương cơ bản} * \text{Hệ số lương} * \text{Hệ số phụ cấp}$$

- Đối với cán bộ hợp đồng:

$$\text{Lương} = \text{Tiền công} * \text{Số ngày công} * \text{Hệ số vượt giờ.}$$

- d. Tính tổng tiền lương mà công ty ABC phải trả cho tất cả các cán bộ.

2. Một trường đại học K cần quản lý danh sách các nhân viên cũng như dựa trên tiêu chí cụ thể để đánh giá lao động tiên tiến. Nhân viên bao gồm 3 loại: giảng viên, nghiên cứu viên và nhân viên văn phòng. Mỗi nhân viên đều có mã nhân viên, họ tên, ngày tháng năm sinh. Với mỗi loại nhân viên, ta có thêm các thông tin khác như sau:

- Giảng viên có thông tin về số tiết giảng dạy trong năm, số đề tài khóa luận mà giảng viên đó hướng dẫn.



- Nghiên cứu viên có thông tin về số đề tài nghiên cứu mà nghiên cứu viên đó tham gia và số bài báo khoa học mà nghiên cứu viên đó là tác giả (hoặc đồng tác giả).
- Nhân viên văn phòng có thông tin ghi nhận số lượng lớp bồi dưỡng chuyên môn mà nhân viên đã tham dự trong năm, số giờ tham gia lao động công ích do trường tổ chức.

Tiêu chuẩn xét tuyển lao động tiên tiến: Giảng viên: giảng dạy ít nhất 300 tiết và hướng dẫn ít nhất 5 khóa luận.

- Nghiên cứu viên, tham gia ít nhất 1 đề tài nghiên cứu và có ít nhất 2 bài báo KH. - Nhân viên VP: tham gia ít nhất 1 lớp bồi dưỡng và ít nhất 20 giờ lao động công ích.

- Xây dựng các lớp (class) và cấu trúc kế thừa phù hợp cho bài toán trên
- Xây dựng lớp quản lý nhân viên (QLNV) cho phép nhập vào danh sách các nhân viên (với loại khác nhau). Sau đó xuất ra các nhân viên đủ tiêu chuẩn lao động tiên tiến. Yêu cầu sử dụng đa hình. Hàm main phải thỏa mãn dạng như sau

```
void main()
{
    QLNv a; // class Quản lý nhân viên
    cin>>a; // Nhập vào danh sách nhân viên
    cout<<a; //In ra danh sách các nhân viên đủ tiêu chuẩn lao động tiên tiến
}
```