Ôn tập
Lập trình hướng đối tượng

Nội dung ôn tập

- Cơ bản về lập trình hướng đối tượng và C++
- Đa năng hóa
- Sự kế thừa
- Bài tập

Cơ bản về hướng đối tượng và C++

Tài liệu tham khảo

- Bài giảng LTHĐT, Trần Minh Châu, Đại học Công nghệ, ĐH Quốc gia HN
- Bài giảng LTHĐT, Nguyễn Ngọc Long, ĐH KHTN TPHCM
- Bài giảng LTHĐT, Huỳnh Lê Tấn Tài, ĐH KHTN TPHCM
- C++ How to Program, Dietel

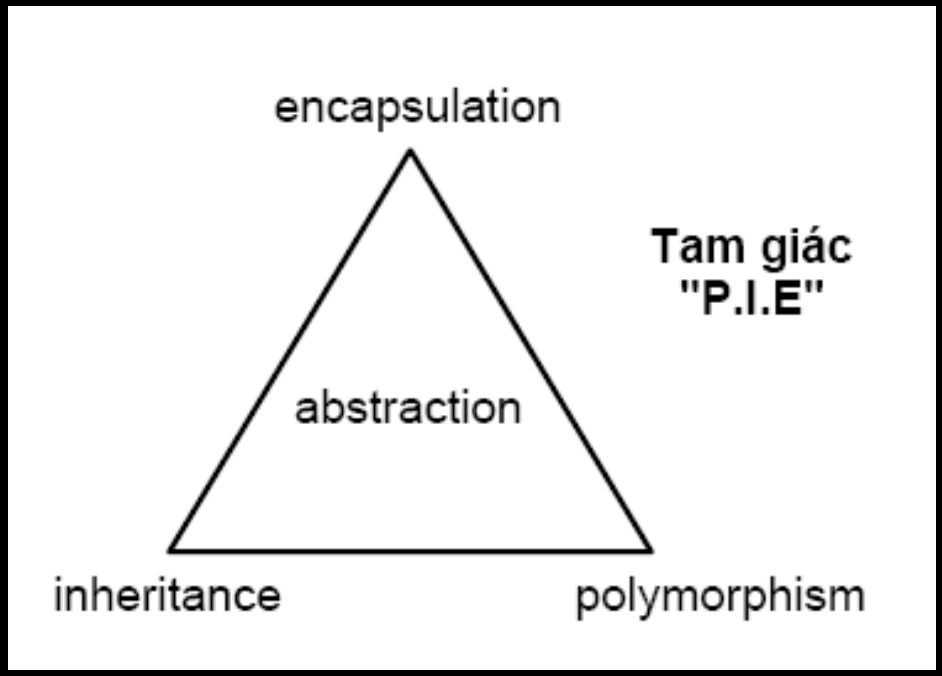
Hướng đối tượng là gì?

- Hiện giờ, đã có sự thống nhất rằng hướng đối tượng là:
  - lớp - class
  - thừa kế - inheritance và liên kết động - dynamic binding

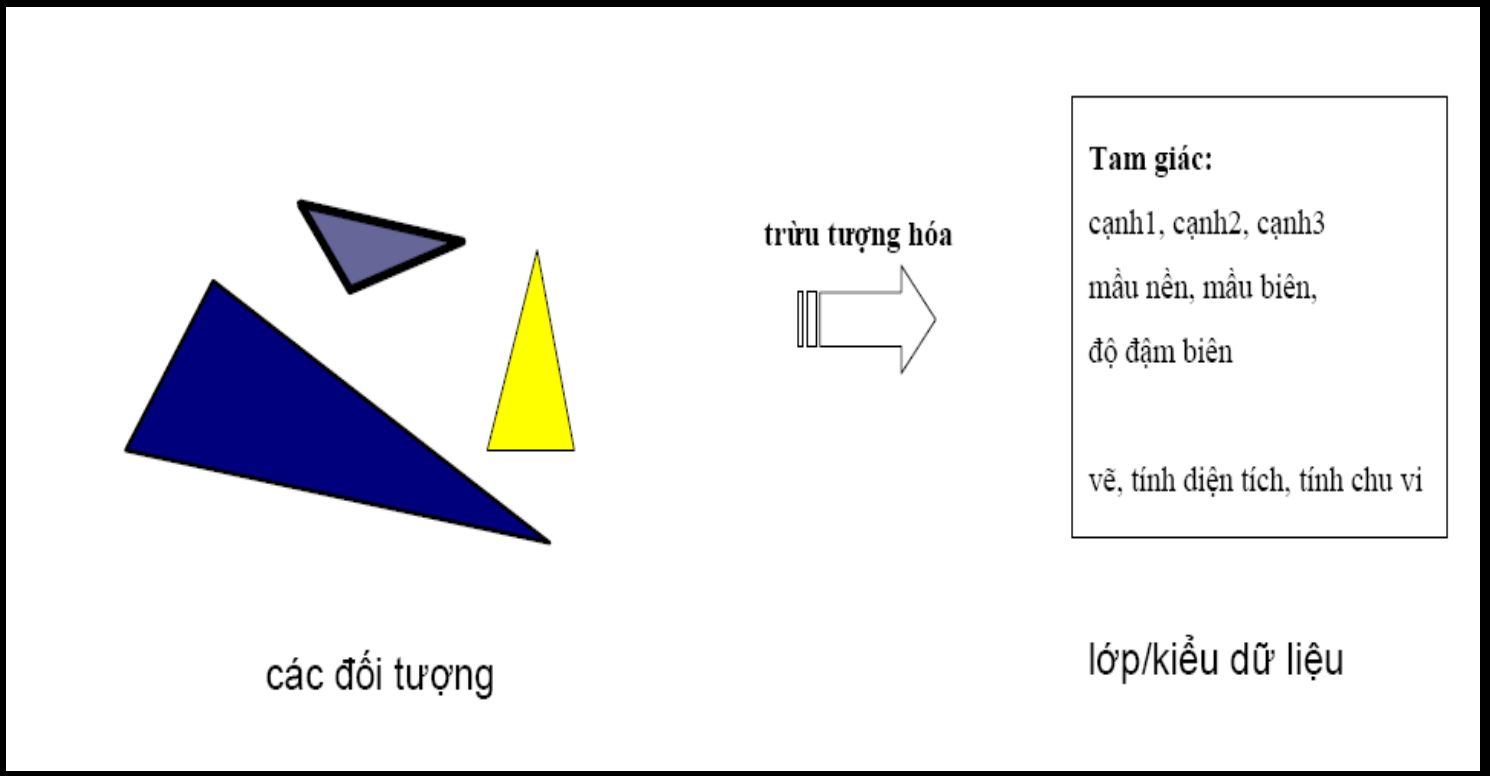
Các đặc điểm quan trọng của hướng đối tượng

- Các lớp đối tượng - Classes
- Đóng gói – Encapsulation
- Thừa kế - Inheritance
- Đa hình - Polymorphism

Các đặc điểm quan trọng của hướng đối tượng



Trừu tượng hóa



cách nhìn **đơn giản hóa** về một đối tượng mà trong đó chỉ bao gồm các đặc điểm được **quan tâm** và bỏ qua những chi tiết không cần thiết.

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

Đóng gói – Che dấu thông tin

- 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

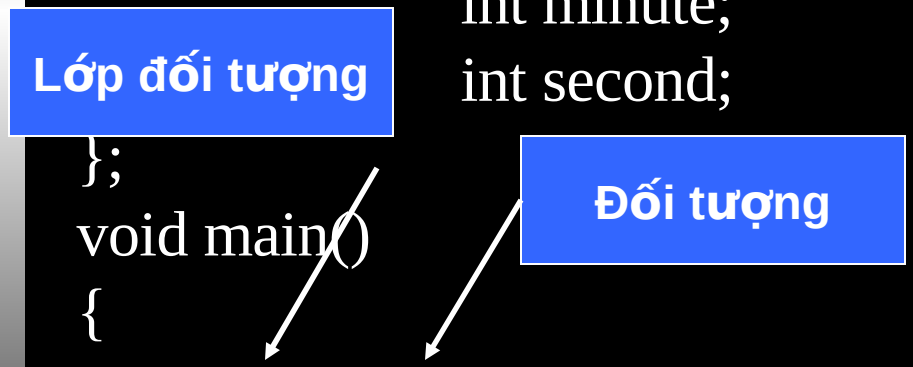
- ***Lưu giữ trạng thái***: mỗi đối tượng có trạng thái (dữ liệu của nó) và các thao tác
- ***Định danh***: Mỗi đối tượng bất kể đang ở trạng thái nào đều có định danh và được đối xử như một thực thể riêng biệt.
- ***Thông điệp***: là phương tiện để một đối tượng A chuyển tới đối tượng B yêu cầu B thực hiện một trong số các thao tác của B.

Lớp đối tượng - class

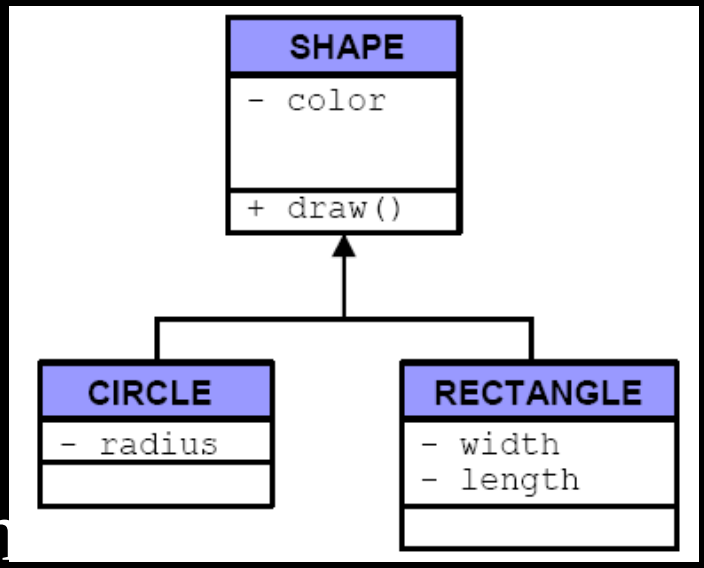
- Lớp: là khuôn mẫu để tạo các đối tượng (tạo các thể hiện). Mỗi đối tượng có cấu trúc và hành vi giống như lớp đối tượng mà nó được tạo từ đó.
- Lớp là cái ta thiết kế và lập trình
- Đối tượng là cái ta tạo (từ một lớp) tại thời gian chạy.

Đối tượng và Lớp đối tượng

```
class Time
{
        private:
                int hour;
                int minute;
                int second;
};
void main()
{
        Time t;
}
```



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
- Đa hình đối tượng - Object polymorphism

Thiết kế hướng đối tượng

- Lập trình hướng đối tượng là quy trình tạo một chương trình dựa theo một thiết kế hướng đối tượng
- Thiết kế hướng đối tượng là quy trình thiết kế một hệ thống sử dụng các nguyên lý thiết kế hướng đối tượng
- C++ được coi là một ngôn ngữ hướng đối tượng vì nó cung cấp các tiện ích đặc biệt cho việc tổ chức chương trình và dữ liệu theo mô hình hướng đối tượng

Khai báo lớp trong C++

- Mô hình đối tượng
  - Thuộc tính (data members)
  - Hành vi (member functions)
- Khai báo với từ khóa **`class`**
- Member functions

Từ khoá xác định phạm vi truy cập

- **`public:`**
  - Truy cập bởi đối tượng của lớp ở bất cứ nơi nào
- **`private:`**
  - Truy cập bởi các hàm thành viên của lớp
- **`protected:`**

Cú pháp

class <tên lớp>:<các lớp cha>{

{

private:

.....

proctected:

.....

public:

.....

};

Class `Time` definition

```
1   class Time {
2
3   public:
4      Time();                    // constructor
5      void setTime( int, int,                    second
6      void printUniversal()                      ormat
7      void printStandard()                       ne format
8
9   private:
10     int hour;     // 0 - 23 (24-hour clock format)
11     int minute;   // 0 - 59
12     int second;   // 0 - 59
13
14  }; // end class Time
```

Nguyên mẫu hàm cuả các hàm thành viên **`public`**

Lớp bắt đầu với từ khóa **`class`**.

Phạm vi truy cập

Hàm khởi tạo

`private` data members

Các hành vi của đối tượng

- Constructor
- Query (truy vấn)
- Update (cập nhật)
- Destructor
- Các hàm chức năng

Constructor

- **Constructor** là một loại phương thức đặc biệt dùng để khởi tạo thể hiện của lớp
- Bất kể loại cấp phát bộ nhớ nào được sử dụng (tự động, tĩnh,động), mỗi khi một thể hiện của lớp được tạo, một hàm constructor nào đó của lớp sẽ được gọi

Constructor

- Constructor **không** có giá trị trả về (kể cả void)
- Constructor có thể được khai báo chồng như các hàm C++ thông thường khác

Constructor mặc định

- **Constructor mặc định (default constructor)** là constructor được gọi khi thể hiện được khai báo mà không có đối số nào được cung cấp
  - MyClass x;
  - MyClass* p = new MyClass;
- Ngược lại, nếu tham số được cung cấp tại khai báo thể hiện, trình biên dịch sẽ gọi phương thức constructor khác (overload)
  - MyClass x(5);
  - MyClass* p = new MyClass(5);

Khai báo các constructor

- Constructor luôn có tên *trùng* với tên lớp
- Do không trả về giá trị, ta khai báo constructor như các phương thức khác nhưng *bỏ qua* kiểu giá trị trả về, cũng có thể cung cấp đối số mặc định.

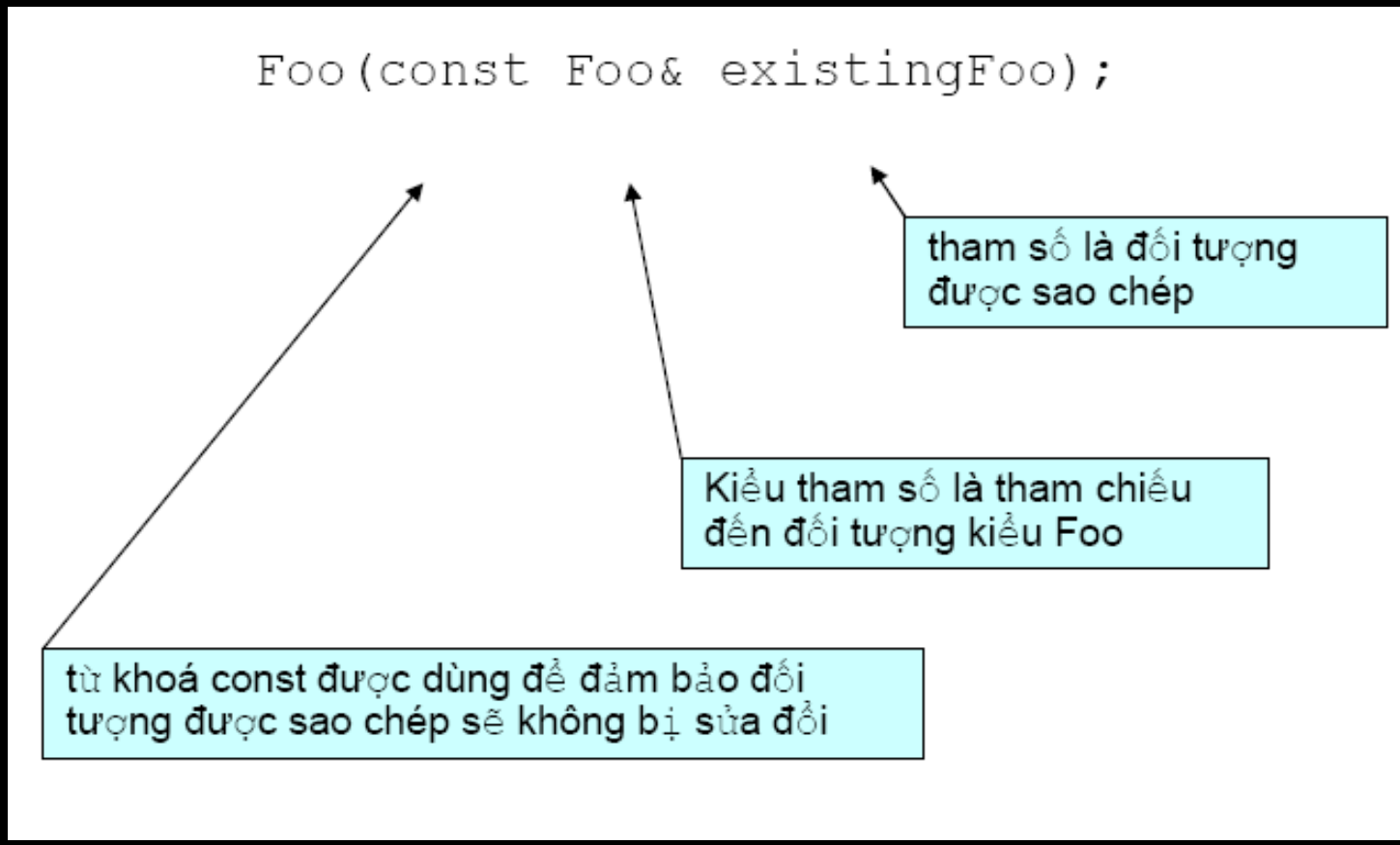
```
class Foo {
public:
    Foo();
    Foo(int x=5);
    Foo(string s);
};
```

Constructor

- Đối với constructor mặc định, nếu ta không cung cấp một phương thức constructor nào, C++ sẽ tự sinh constructor mặc định là một phương thức rỗng
- Tuy nhiên, nếu ta không định nghĩa constructor mặc định nhưng lại có các constructor khác, trình biên dịch sẽ báo lỗi không tìm thấy constructor mặc định nếu ta không cung cấp tham số khi tạo thể hiện.

Copy constructor

```
Foo(const Foo& existingFoo);
```



Destructor

- Cũng như một phương thức constructor được gọi khi một đối tượng được tạo, loại phương thức thứ hai, destructor, được gọi ngay trước khi **thu hồi một đối tượng**
- Destructor thường được dùng để thực hiện mọi việc dọn dẹp cần thiết trước khi một đối tượng bị huỷ
- Destructor không có giá trị trả về, và không thể định nghĩa lại (nó không bao giờ có tham số)
- Phương thức destructor trùng tên với tên lớp nhưng có dấu ~ đặt trước

Cập nhật

```
int Student::setGPA(double newGPA)
{
        if ((newGPA >= 0.0) && (newGPA <= 10.0))
        {
                this->gpa = newGPA;
                return 0; // Return 0 to indicate success
        }
        else
        {
                return -1; // Return -1 to indicate failure
        }
}
```

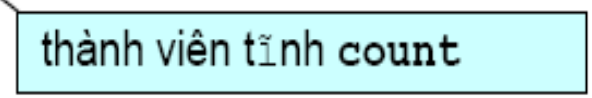
Thành viên tĩnh - Ví dụ

- Đếm số đối tượng MyClass

```
class MyClass {
   public:
      MyClass(); // Constructor
      ~MyClass(); // Destructor

      void printCount();  // Output current value of count

   private:
      static int count;   // static member to store
                          //    number of instances of MyClass
};
```



Khởi tạo biến đếm bằng 0, vì ban đầu không có đối tượng nào

```
int MyClass::count = 0;

MyClass::MyClass() {
   this->count++; // Increment the static count
}

MyClass::~MyClass() {
   this->count--; // Decrement the static count
}

void MyClass::printCount() {
   cout << "There are currently " << this->count
        << " instance(s) of MyClass.\n";
}
```

```
int main()
{
    MyClass* x = new MyClass;
    x->PrintCount();

    MyClass* y = new MyClass;
    x->PrintCount();
    y->PrintCount();

    delete x;
    y->PrintCount();
}
```

```
There are currently 1 instance(s) of MyClass.
There are currently 2 instance(s) of MyClass.
There are currently 2 instance(s) of MyClass.
There are currently 1 instance(s) of MyClass.
```

Hằng phương thức – const method

- Từ khoá **const** được dùng cho các tham số của hàm để đảm bảo các tham số được truyền cho hàm sẽ không bị hàm sửa đổi.
  - int myFunction(**const** int& x);
- Còn tham số ẩn truyền bằng con trỏ this và chính là đối tượng chủ?

```
class MyClass {
    ...
    void printCount() const;
    ...
};
```

```
...
void MyClass::PrintCount() const
{
  // ...
}
```

phải có từ khóa **const** ở cả khai báo và định nghĩa phương thức

Phương thức thiết lập với giá trị mặc định

```
class Diem
{
public:
                                    double x,y;

                                    Diem(double xx = 0, double yy =
                                    0):x(xx), y(yy){}
                                    void Set(double xx, double yy) {x =
                                    xx, y = yy;}
                                    // ...
};
class String
{
public:
                                    char *p;

                                    String(char *s = “”) {p = strdup(s);}
                                    String(const String &s) {p =
                                    strdup(s.p);}
                                    ~String() {cout << "delete "<< (void
```

Phương thức thiết lập với giá trị mặc định

```
class SinhVien
{
                    String MaSo;
                    String HoTen;
                    int NamSinh;
public:
                    SinhVien(char *ht = “Nguyen Van A”, char *ms =
                    “19920014”, int ns = 1982):HoTen(ht), MaSo(ms),
                    NamSinh(ns){}
                    //...
};

String as[3]; // Ok: Ca ba phan tu deu la chuoi rong
Diem ad[5];          // Ok: ca 5 diem deu la (0,0)
SinhVien asv[7];     // Ok: Het sai ca 7 sinh vien deu co cung hoten, maso,
                     namsinh
```

Phương thức thiết lập không tham số

```
class Diem
{

public:

                                    double x,y;

                                    Diem(double xx, double yy):x(xx),
                                    y(yy){}
                                    Diem():x(0), y(0){}
                                    // ...

};
class String
{

public:

                                    char *p;

                                    String(char *s) {p = strdup(s);}
                                    String() {p = strdup(“”);}
                                    ~String() {cout << "delete "<< (void
                                    *)p << "\n"; delete [] p;}
                                    // ...

};
```

Phương thức thiết lập không tham số

```
class SinhVien {
                    String MaSo;
                    String HoTen;
                    int NamSinh;
public:
                    SinhVien(char *ht, char *ms, int ns):HoTen(ht),
                    MaSo(ms), NamSinh(ns){}
                    SinhVien():HoTen("Nguyen Van A"),
                    MaSo("19920014"), NamSinh(1982){}
                    //...
};
String as[3]; // Ca ba phan tu deu la chuoi rong
Diem ad[5];         // ca 5 diem deu la (0,0)
SinhVien asv[7];// Ca 7 sinh vien deu co cung hoten, maso, namsinh
```

Đối tượng được cấp phát động

```
class String {
    char *p;
public:
    String(char *s) {p = strdup(s);}
    String(const String &s) {p = strdup(s.p);}
    ~String() {delete [] p;}
    //...
};
class Diem {
    double x,y;
public:
    Diem(double xx, double yy):x(xx),y(yy){};
    //...
};
//...
```

Cấp và hủy một đối tượng

```
int *pi = new int;
int *pj = new int(15);
Diem *pd = new Diem(20,40);
String *pa = new String("Nguyen Van A");
//...
delete pa;
delete pd;
delete pj;
delete pi;
```

Cấp và hủy nhiều đối tượng

```
int *pai = new int[10];
Diem *pad = new Diem[5];
      // ca 5 diem co cung toa do (0,0)
String *pas = new String[5];
// Ca 5 chuoi cung duoc khoi dong bang “Alibaba”


delete [] pas;
delete [] pad;
delete [] pai;
```

Lớp ThoiDiem – Cách 1

```
class ThoiDiem
{
                int gio, phut, giay;
                static bool HopLe(int g, int p, int gy);
public:
                ThoiDiem(int g = 0, int p = 0, int gy = 0) {Set(g,p,gy);}
                void Set(int g, int p, int gy);
                int LayGio() const {return gio;}
                int LayPhut() const {return phut;}
                int LayGiay() const {return giay;}
                void Nhap();
                void Xuat() const;
                void Tang();
                void Giam();
};
```

Lớp ThoiDiem – Cách 2

```
class ThoiDiem
{
                long tsgiay;
                static bool HopLe(int g, int p, int gy);
public:
                ThoiDiem(int g = 0, int p = 0, int gy = 0) {Set(g,p,gy);}
                void Set(int g, int p, int gy);
                int LayGio() const {return tsgiay / 3600;}
                int LayPhut() const {return (tsgiay%3600)/60;}
                int LayGiay() const {return tsgiay % 60;}
                void Nhap();
                void Xuat() const;
                void Tang();
                void Giam();
};
```

Đa năng hóa toán tử

Các toán tử có thể đa năng hóa

+	-	*	/	%
^ &	|	!	=	<
>	+=	-=	*=	/=
~=	%=	^=	&=	|=
>>=	<<=	==	!=	<=
>=	&&	||	++	--
,	->	->*	()	[]
new	delete	new[]	delete[]	

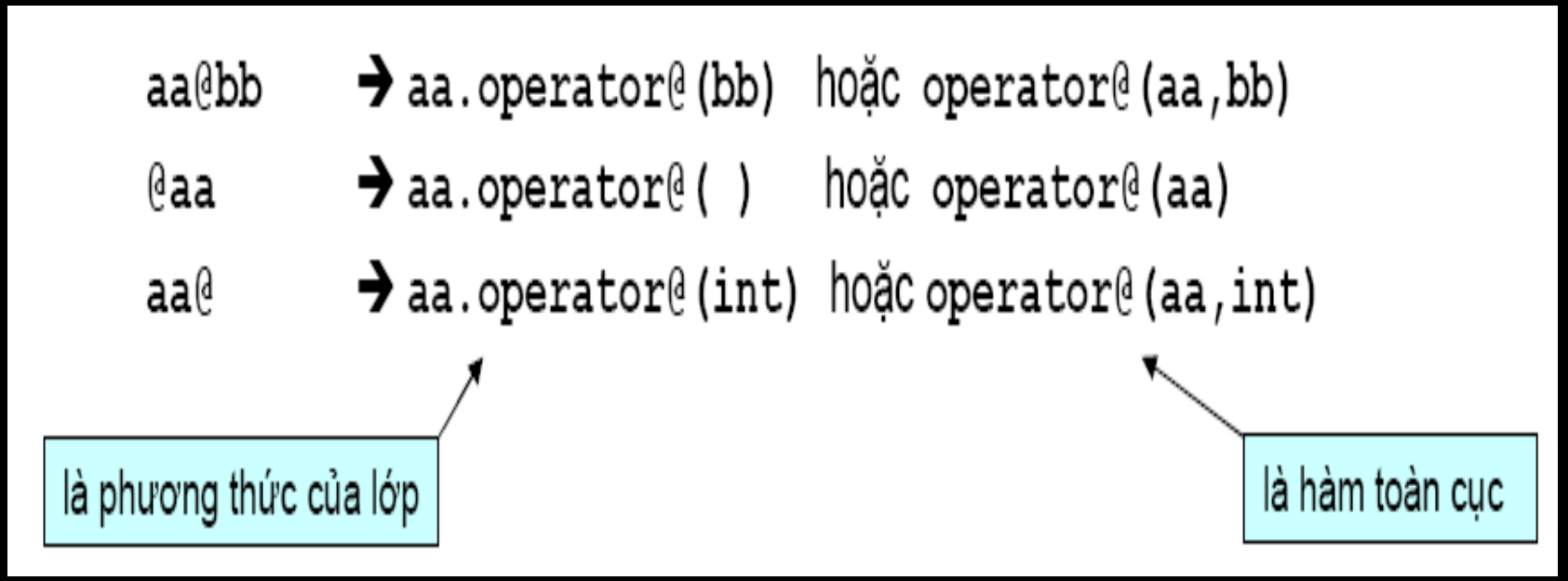
Các toán tử không thể đa năng hóa

`.`	`.*`
`::`	`:?`
`typeid`	`sizeof`
`const_cast`	`dynamic_cast`
`reinterpret_cast`	`static_cast`

Cú pháp của Operator Overloading

- Khai báo và định nghĩa toán tử thực chất không khác với việc khai báo và định nghĩa một loại hàm bất kỳ nào khác
- Sử dụng tên hàm là "operator@" cho toán tử "@“: operator+
- Số lượng tham số tại khai báo phụ thuộc hai yếu tố:
  - Toán tử là toán tử đơn hay đôi
  - Toán tử được khai báo là hàm toàn cục hay phương thức của lớp

Cú pháp của Operator Overloading



Lớp PhanSo

```
class PhanSo
{
    long tu, mau;
    void UocLuoc();
public:
    PhanSo(long t, long m) {Set(t,m);}
    void Set(long t, long m);
    long LayTu() const {return tu;}
    long LayMau() const {return mau;}
    PhanSo Cong(PhanSo b) const;
    PhanSo operator + (PhanSo b) const;
    PhanSo operator - () const {
        return PhanSo(-tu, mau);
    }
    bool operator == (PhanSo b) const;
    bool operator != (PhanSo b) const;
    void Xuat() const;
};
```

Lớp PhanSo

```
PhanSo PhanSo::Cong(PhanSo b) const {
    return PhanSo(tu*b.mau + mau*b.tu, mau*b.mau);
}
PhanSo PhanSo::operator + (PhanSo b) const {
    return PhanSo(tu*b.mau + mau*b.tu, mau*b.mau);
}
bool PhanSo::operator == (PhanSo b) const {
    return tu*b.mau == mau*b.tu;
}
void PhanSo::Xuat() const {
    cout << tu;
    if (tu != 0 && mau != 1)
        cout << "/" << mau;
}
```

Ví dụ đa năng hóa toán tử

```
class PhanSo {
    long tu, mau;
    void UocLuoc();
public:
    PhanSo(long t, long m) {Set(t,m);}
    void Set(long t, long m);
    long LayTu() const {return tu;}
    long LayMau() const {return mau;}
    PhanSo operator + (PhanSo b) const;
    friend PhanSo operator - (PhanSo a, PhanSo b);
    PhanSo operator -() const {return PhanSo(-tu, mau);}
    bool operator == (PhanSo b) const;
    bool operator != (PhanSo b) const;
    void Xuat() const;
};
```

Ví dụ đa năng hóa toán tử

```
PhanSo PhanSo::operator + (PhanSo b) const {
    return PhanSo(tu*b.mau + mau*b.tu, mau*b.mau);
}
PhanSo operator - (PhanSo a, PhanSo b) {
    return PhanSo(a.tu*b.mau - a.mau*b.tu, a.mau*b.mau);
}

void main() {
    PhanSo a(2,3), b(3,4), c(0,1),d(0,1);
    c = a + b;          // d = a.operator + (b);
    d = a - b;          // d = operator - (a,b);
    cout << "c = "; c.Xuat(); cout << "\n";
    cout << "d = "; d.Xuat(); cout << "\n";
}
```

Đa năng hóa toán tử << và >>

```
//phanso.h
class PhanSo {
    long tu, mau;
    void UocLuoc();
public:
    PhanSo(long t = 0, long m = 1) {Set(t,m);}
    void Set(long t, long m);
    long LayTu() const {return tu;}
    long LayMau() const {return mau;}
    friend PhanSo operator + (PhanSo a, PhanSo b);
    friend PhanSo operator - (PhanSo a, PhanSo b);
    friend PhanSo operator * (PhanSo a, PhanSo b);
    friend PhanSo operator / (PhanSo a, PhanSo b);
    PhanSo operator -() const {return PhanSo(-tu,mau);}
    friend istream& operator >> (istream &is, PhanSo &p);
    friend ostream& operator << (ostream &os, PhanSo p);
};
```

Đa năng hóa toán tử << và >>

```
// phanso.cpp
#include <iostream.h>
#include “phanso.h”
istream & operator >> (istream &is, PhanSo &p)
{
    is >> p.tu >> p.mau;
    while (!p.mau)
    {
            cout << “Nhap lai mau so: ”;
            is >> p.mau;
    }
    p.UocLuoc();
    return is;
}
ostream & operator << (ostream &os, PhanSo p)
{
    os << p.tu;
    if (p.tu != 0 && p.mau != 1)
            os << "/" << p.mau;
    return os;
}
```

Đa năng hóa toán tử << và >>

```
// tps.cpp
#include <iostream.h>
#include “phanso.h”
void main()
{
    PhanSo a, b;
    cout << “Nhap phan so a: ”; cin >> a;
    cout << “Nhap phan so b: ”; cin >> b;
    cout << a << " + " << b << " = " << a + b << "\n";
    cout << a << " - " << b << " = " << a - b << "\n";
    cout << a << " * " << b << " = " << a * b << "\n";
    cout << a << " / " << b << " = " << a / b << "\n";
}
```

Đa năng hóa toán tử ++ và --

```
class ThoiDiem
{
    long tsgiay;
    static bool HopLe(int g, int p, int gy);
public:
    ThoiDiem(int g = 0, int p = 0, int gy = 0);
    void Set(int g, int p, int gy);
    int LayGio() const {return tsgiay / 3600;}
    int LayPhut() const {return (tsgiay%3600)/60;}
    int LayGiay() const {return tsgiay % 60;}
    void Tang();
    void Giam();
    ThoiDiem &operator ++();
};
```

Đa năng hóa toán tử ++ và --

```
void ThoiDiem::Tang()
{
	tsgiay = ++tsgiay%SOGIAY_NGAY;
}
void ThoiDiem::Giam()
{
	if (--tsgiay < 0) tsgiay = SOGIAY_NGAY-1;
}
ThoiDiem &ThoiDiem::operator ++()
{
	Tang();
	return *this;
}
```

Đa năng hóa toán tử ++ và --

```
void main() {
  clrscr();
        ThoiDiem t(23,59,59),t1,t2;
        cout << "t = " << t << "\n";
        t1 = ++t; // t.operator ++();
                // t = 0:00:00, t1 = 0:00:00
        cout << "t = " << t << "\tt1 = " << t1 << "\n";
        t1 = t++; // t.operator ++();
                // t = 0:00:01, t1 = 0:00:00
        cout << "t = " << t << "\tt1 = " << t1 << "\n";
}
```

Sự kế thừa

Kế thừa đơn

```
class Nguoi {
   char *HoTen;
   int NamSinh;
public:
   Nguoi(char *ht, int ns):NamSinh(ns)
   {HoTen=strdup(ht);}
   ~Nguoi() {delete [] HoTen;}
   void An() const { cout<<HoTen<<" an 3 chen
   com";}
   void Ngu() const { cout<<HoTen<< " ngu ngay 8
   tieng";}
   void Xuat() const;
   friend ostream& operator << (ostream &os,
   Nguoi& p);
};
```

Kế thừa đơn

```
class SinhVien : public Nguoi {
   char *MaSo;
public:
   SinhVien(char *ht, char *ms, int ns) :
   Nguoi(ht,ns) {  MaSo = strdup(ms);
   }
   ~SinhVien() {delete [] MaSo;}
   void Xuat() const;
};
ostream& operator << (ostream &os, Nguoi& p)
{
   return os << "Nguoi, ho ten: " << p.HoTen << "
   sinh " << p.NamSinh;
}
```

Kế thừa đơn

```
void Nguoi::Xuat() const {
   cout << "Nguoi, ho ten: " << HoTen
        << " sinh " << NamSinh;
}

void SinhVien::Xuat() const {
   cout << "Sinh vien, ma so: " << MaSo
        << ", ho ten: " << HoTen;
}
```

Kế thừa đơn

```
void main() {
   Nguoi p1("Le Van Nhan",1980);
   SinhVien s1("Vo Vien Sinh", "200002541",1984);
   cout << ”1.\n";
   p1.An(); cout << "\n";
   s1.An();cout << "\n";
   cout << ”2.\n";
   p1.Xuat(); cout << "\n";
   s1.Xuat(); cout << "\n";
   s1.Nguoi::Xuat(); cout << "\n";
   cout << "3.\n";
   cout << p1 << "\n";
   cout << s1 << "\n";
}
```

Phạm vi truy xuất

Hình thức trong lớp cơ sở	Hình thức kế thừa	Hình thức truy cập trong lớp dẫn xuất
private private private	private protected public	Không thể truy cập Không thể truy cập Không thể truy cập
protected protected protected	private protected public	private protected protected
public public public	private protected public	private protected public

Bài tập

- Xây dựng các lớp đối tượng
  - Dãy, danh sách liên kết
  - Tập hợp
  - Phân số, số phức
  - Tam giác, tứ giác
  - Ngăn xếp, hàng đợi
  - Đơn thức, đa thức

Đề thi

- 1 câu lý thuyết ~ 0.5 - 0.75 d
- 1 câu bài tập ~ 2.25 – 2.5 d
- Thời gian làm bài: 60’