

# Lập trình hướng đối tượng và C++

## Bài 4: Đối tượng và lớp

**TS. Nguyễn Hiếu Cường**

Bộ môn CNPM, Khoa CNTT,

Trường Đại học GTVT

Email: [cuonggt@gmail.com](mailto:cuonggt@gmail.com)

# Nội dung chính

---

1. Giới thiệu môn học
2. Các khái niệm cơ bản
3. Hàm trong C++
- 4. Đối tượng và lớp**
5. Định nghĩa chồng toán tử
6. Hàm tạo và hàm huỷ
7. Dẫn xuất và thừa kế
8. Tương ứng bội
9. Khuôn hình (templates)

# Đối tượng và lớp

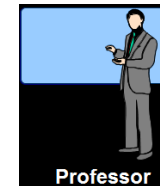
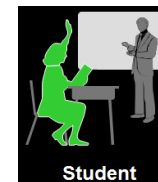
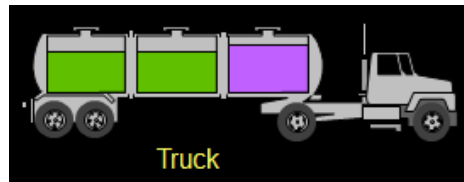
---

- Khái niệm về đối tượng và lớp
- Khai báo và định nghĩa lớp
- Các thành phần của lớp
- Biến đối tượng và con trỏ đối tượng
- Con trỏ *this*
- Đối tượng là đối của hàm
- Hàm bạn và lớp bạn

# Đối tượng

- Thế giới thực bao gồm các *đối tượng* (object)!

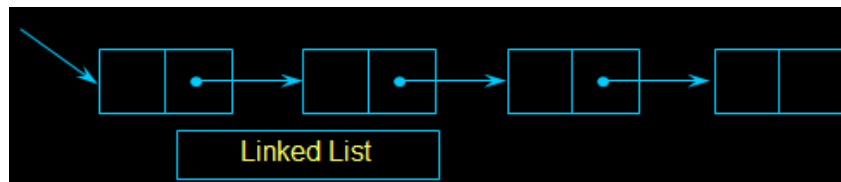
- Đối tượng vật lý



- Đối tượng khái niệm



- Đối tượng phần mềm



- Mỗi đối tượng gồm các *thuộc tính* và các *thao tác*.

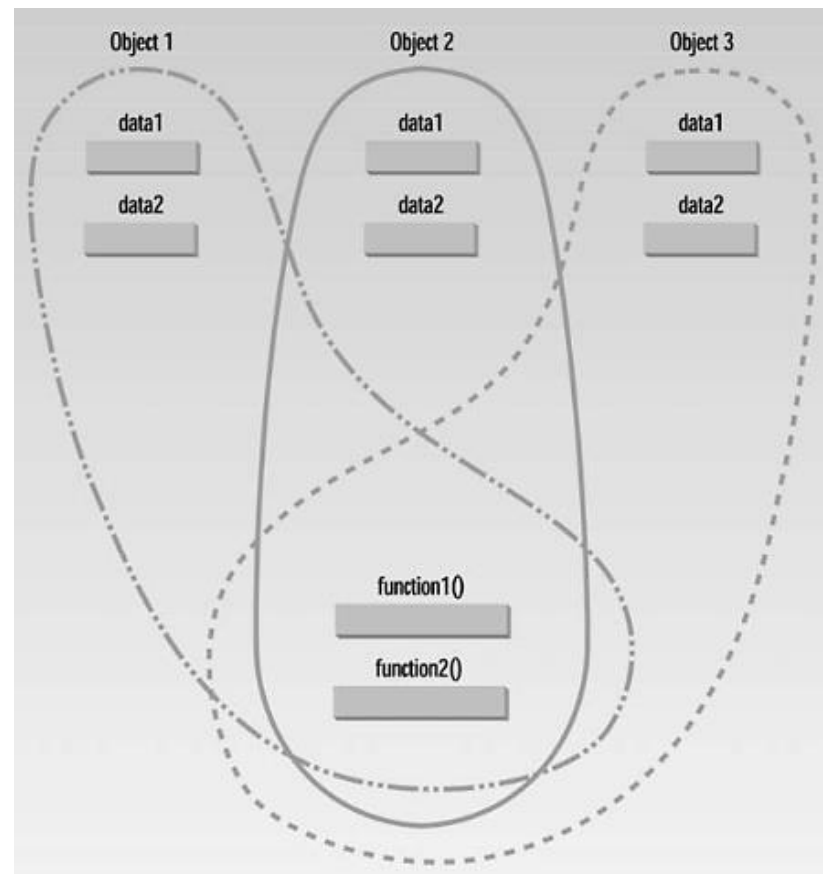
# Lớp

---

- Lớp (class): định nghĩa trừu tượng (abstract definition) của các đối tượng có cùng những đặc tính chung
- Đối tượng (object): thể hiện cụ thể (instance) của một lớp
- Tác dụng của lớp?
  - Trừu tượng hoá dữ liệu (data abstraction)
  - Bao gói (encapsulation): dữ liệu + thao tác
  - Che giấu thông tin (information hiding)
- Liên hệ với các khái niệm đã biết:
  - Lớp  $\approx$  Kiểu
  - Đối tượng  $\approx$  Biến

# Đối tượng, lớp và bộ nhớ

- Mỗi đối tượng của một lớp là độc lập với nhau (dữ liệu riêng)
  - Dữ liệu của từng đối tượng được cấp bộ nhớ mỗi khi đối tượng được tạo ra
- Các đối tượng của cùng lớp sử dụng chung các hàm thành phần
  - Hàm thành phần được tạo và đặt trong bộ nhớ *một lần* khi định nghĩa lớp
  - Hàm thành phần của lớp được chia sẻ giữa các đối tượng của lớp
  - Không có xung đột vì mỗi thời điểm chỉ có một hàm được thực hiện



# Khai báo lớp

---

- Lớp cần khai báo trước khi sử dụng

```
// Khai báo lớp
class tên_lớp
{
[private:]
    // Khai báo các thành phần dữ liệu
public:
    // Khai báo các phương thức
};
```

# Các thành phần của lớp

---

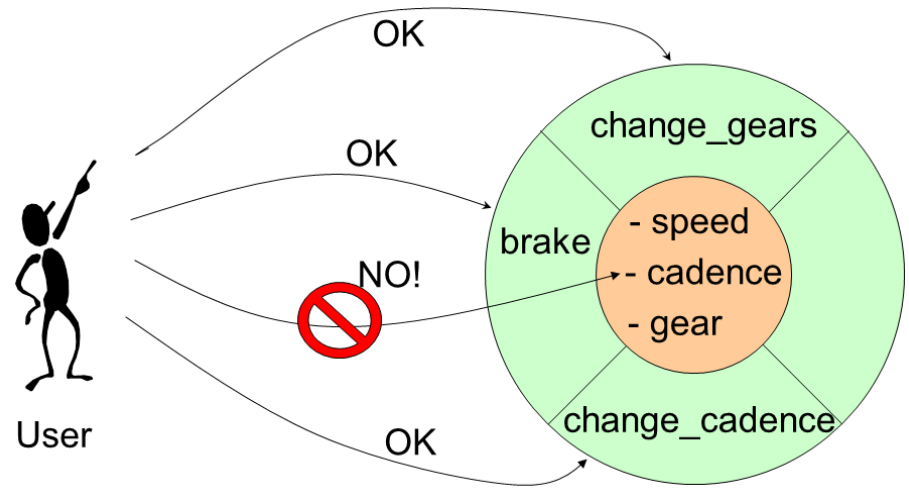
- Mỗi thành phần của lớp có phạm vi truy cập nhất định
- Phạm vi truy cập xác định bởi các từ khóa:
  - **private** [mặc định]  
Không thể truy nhập trực tiếp từ bên ngoài lớp, mà phải thông qua các phương thức của lớp
  - **public**  
Có thể truy nhập trực tiếp từ bên ngoài lớp
  - **protected**  
Tương tự như private, ngoại trừ với các lớp dẫn xuất



# Ví dụ (lớp)

```
class Bicycle
{
private:
    float speed;
    float cadence;
    int gear;
public:
    void change_gears(int gear);
    void brake();
    void chage_cadence(float cadence);
};

...
Bicycle b;
b.brake();
b.cadence = 5;           // Error
b.change_cadence(5);      // OK
```



# Ví dụ

- Kết quả của chương trình?

```
class A {  
public:  
    int x;  
};  
  
int main() {  
    A a;  
    a.x = 50;  
    cout<<a.x;  
}
```

- Chương trình có lỗi, tại sao?

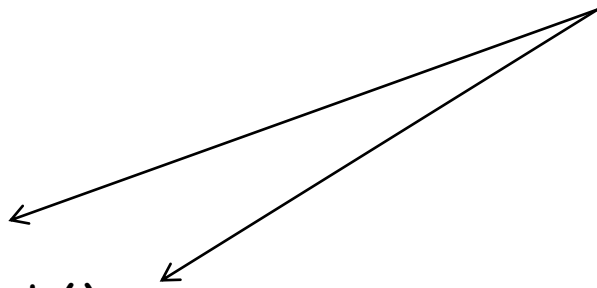
```
class A {  
  
    int x;  
};  
  
int main() {  
    A a;  
    a.x = 50;  
    cout<<a.x;  
}
```

# Phạm vi của các thành phần trong lớp

- Dữ liệu thường là riêng (private) để đảm bảo tính bảo mật
- Để truy nhập các thành phần riêng của một lớp cần thông qua các phương thức của lớp đó

```
class A {  
    int x;  
public:  
    void nhap() { cin>>x; }  
    void xuat() { cout<<x; }  
};  
int main() {  
    A a;  
    a.nhap();  
    cout<<a.xuat();  
}
```

Truyền thông điệp (message passing)  
nhap() và xuat() cho đối tượng a



# Biến đối tượng

---

- Khai báo biến đối tượng

`Tên_lớp Tên_biến ;`

- Truy nhập vào thành phần của đối tượng

`Tên_biến . Tên_thành_phần`

- Truyền thông điệp cho một đối tượng

`A x;`

`x.nhap(); // truyền thông điệp nhap() cho đối tượng a`

# Con trỏ đối tượng

---

- Con trỏ đối tượng chứa địa chỉ của biến đối tượng
- Sử dụng để truy nhập vào các thành phần đối tượng
- Khai báo con trỏ đối tượng

```
Tên_lớp *Tên_con_trỏ;
```

- Sử dụng con trỏ đối tượng

```
Tên_con_trỏ -> Tên_thành_phần
```

```
A x, *p;  
x.nhap();  
p = &x;  
p -> xuat();
```

# Ví dụ

---

```
#include <iostream>
using namespace std;

// Khai bao lop
class DIEM
{
    int x, y;
public:
    void an();
    void nhapsl();
    void hien();
};

...
```

```
int main()
{
    DIEM *p;
    int i, n;
    cout << "So diem: ";
    cin >> n;
    p = new DIEM[n+1];
    for (i=1; i<=n; ++i)
        p[i] -> nhapsl();
    for (i=1; i<=n; ++i)
        p[i] -> hien();
    for (i=1; i<=n; ++i)
        p[i] -> an();
}
```

# Ví dụ (lớp PS)

---

- Xây dựng lớp Phân số (PS) có
  - Các thành phần dữ liệu ts, ms
  - Các phương thức nhap(), xuat() và nhanps()
  - Không cần tối giản phân số

[B2\_phanso.cpp]

# Con trỏ *this*

---

- C++ cung cấp một từ khóa 'this' để thể hiện đối tượng hiện tại (the current object) và được truyền như một đối ẩn tới các hàm thành phần của lớp
- Con trỏ this chứa địa chỉ bộ nhớ của đối tượng hiện tại
- Con trỏ this thường không được thể hiện tường minh

```
void PS::nhap()  
{  
    cout <<"Nhap phan so: ";  
    cin >> ts >> ms;        // cin >> this-> ts >> this-> ms;  
}
```



# Ví dụ

```
#include <iostream>
using namespace std;
class Test {
    int x;
public:
    Test(int x1 = 0 ) { x = x1; }
    void print();
};
void Test::print() {
    cout << "          x = " << x << endl;
    cout << "  this->x = " << this->x << endl;
    cout << "(*this).x = " << (*this).x;
}
int main()
{
    Test testObject(12); // gọi hàm tạo 1 đối
    testObject.print();
}
```

x	= 12
this->x	= 12
(*this).x	= 12

# Ví dụ

```
#include <iostream>
using namespace std;
class DIEM
{
private:
    int x, y ;
public:
    void nhapsl() { ... }
    void ve_dt(DIEM d2, int mau);
    void ve_tg(DIEM d2, DIEM d3, int
mau);
};
void DIEM::ve_dt(DIEM d2, int mau)
{
    setcolor(mau);
    line(x, y, d2.x, d2.y);
}

void DIEM::ve_tg(DIEM d2, DIEM
d3,int mau)
{
    (*this).ve_dt(d2,mau);
    d2.ve_dt(d3,mau);
    d3.ve_dt(*this,mau);
}
int main()
{
    DIEM d1, d2, d3;
    d1.nhapsl();
    d2.nhapsl();
    d3.nhapsl();
    d1.ve_tg(d2,d3,15);
}
```

# Ví dụ

---

Hãy xây dựng lớp Phân số (PS) có:

- Các thành phần dữ liệu: ts, ms
- Các phương thức nhập(), xuất() [dạng ts/ms]
- Phương thức nhan() để nhân hai phân số
- Phương thức cong() để cộng hai phân số

Xây dựng hàm main() trong đó sử dụng lớp PS

# Đối tượng là đối của hàm

```
#include <iostream>
using namespace std;
////////////////////////////////////
class Distance                      //English Distance class
{
private:
    int feet;
    float inches;
public:
    //constructor (no args)
    Distance() : feet(0), inches(0.0)
    { }

    //constructor (two args)
    Distance(int ft, float in) : feet(ft), inches(in)
    { }

    void getdist()                  //get length from user
    {
        cout << "\nEnter feet: "; cin >> feet;
        cout << "Enter inches: "; cin >> inches;
    }

    void showdist()                  //display distance
    { cout << feet << "'-" << inches << "'"; }

    void add_dist( Distance, Distance ); //declaration
};
//-----
//add lengths d2 and d3
void Distance::add_dist(Distance d2, Distance d3)
{
    inches = d2.inches + d3.inches; //add the inches
    feet = 0;                        //(for possible carry)
    if(inches >= 12.0)                //if total exceeds 12.0,
    {                                //then decrease inches
        inches -= 12.0;              //by 12.0 and
        feet++;                      //increase feet
    }                                //by 1
    feet += d2.feet + d3.feet;       //add the feet
}
////////////////////////////////////
int main()
{
    Distance dist1, dist3;           //define two lengths
    Distance dist2(11, 6.25);        //define and initialize dist2

    dist1.getdist();                 //get dist1 from user
    dist3.add_dist(dist1, dist2);    //dist3 = dist1 + dist2

    //display all lengths
    cout << "\ndist1 = "; dist1.showdist();
    cout << "\ndist2 = "; dist2.showdist();
    cout << "\ndist3 = "; dist3.showdist();
    cout << endl;
    return 0;
}

Enter feet: 17
Enter inches: 5.75

dist1 = 17'-5.75"
dist2 = 11'-6.25"
dist3 = 29'-0"
```

# Ví dụ

---

Xây dựng lớp Điểm, gồm:

- Dữ liệu (private): hoành độ, tung độ
- Các phương thức (public):
  - Nhập điểm
  - Xuất điểm ra màn hình theo dạng (hoành độ, tung độ)
  - Tính khoảng cách giữa hai điểm

Xây dựng hàm main() để sử dụng lớp trên.

# Hàm bạn

---

- Vấn đề
  - Mỗi hàm thành phần (phương thức) chỉ có thể truy nhập vào thành phần riêng của lớp mà nó thuộc vào
  - Làm sao truy nhập vào thành phần riêng của nhiều lớp?
- Giải pháp: Dùng **hàm bạn** (friend function)
  - Là một hàm độc lập, không phải phương thức của lớp
  - Có thể truy nhập vào các thành phần riêng của lớp mà nó làm bạn

```
class A
{
    friend void f(...);    // f là bạn của lớp A
};
```

# Hàm bạn của nhiều lớp

---

Một hàm có thể đồng thời là bạn của nhiều lớp

```
class A
{
    friend    void    f(...); // f là bạn của lớp A
};
```

```
class B
{
    friend    void    f(...); // f là bạn của lớp B
};
```

# Ví dụ

```
// cộng hai số phức là phương thức
class SP {
    double a;
    double b;
public:
    SP cong(SP u2);
};
SP SP::cong(SP u2) {
    SP t;
    t.a = a + u2.a;
    t.b = b + u2.b;
    return t;
}
int main() {
    SP u, u1, u2;
    u = u1.cong(u2);
    ...
}
```

```
// cộng hai số phức dùng hàm bạn
class SP {
    double a;
    double b;
public:
    friend SP cong(SP u1, SP u2);
};
SP cong(SP u1, SP u2) {
    SP t;
    t.a = u1.a + u2.a;
    t.b = u1.b + u2.b;
    return t;
}
int main() {
    SP u, u1, u2;
    u = cong(u1, u2);
    ...
}
```



# Ví dụ (hàm bạn của nhiều lớp)

```
#include <iostream>
using namespace std;
class B;
class A
{
    float X;
public:
    A() { X = 5.0; }
    friend float Init(A,B);
};
class B
{
    float Y;
public:
    B() { Y = 1.0; }
    friend float Init(A,B);
};
```

```
float Init(A a, B b)
{
    return a.X + b.Y;
}

int main()
{
    A a;
    B b;
    cout << Init(a, b);
}

// Kết quả = ?
// 6
```

# Lớp bạn

---

- Lớp A là bạn của lớp B: Các phương thức của lớp A đều là bạn của lớp B

```
class B
{
    ...
    friend class A ;// Lớp A là bạn của B nên:
                    // các phương thức của A đều là bạn của B
};
```

- Các tính chất của lớp bạn:
  - Một lớp có thể là **bạn của nhiều lớp** khác nhau
  - **Không xđng**: A là bạn của B không có nghĩa B cũng là bạn của A
  - **Không bắc cầu**: A là bạn của B, B là bạn của C không có nghĩa A là bạn của C

# Các lớp là bạn của nhau

---

Để khai báo lớp này là bạn của lớp kia, ta viết theo mẫu sau:

```
class B ;  
class A  
{  
    ...  
    friend class B ; // Lớp B là bạn của A  
};  
class B  
{  
    ...  
    friend class A ; // Lớp A là bạn của B  
};
```

# Ví dụ

---

- Lớp bạn có thể truy nhập vào thành phần riêng của lớp mà nó làm bạn
- Ví dụ: lớp LinkedList là bạn của lớp Node → có thể truy nhập vào các thành phần riêng của lớp Node

```
class Node
{
private:
    int key;
    Node *next;
    /* Other members of Node Class */
    ...
    friend class LinkedList; // Now class LinkedList can
                             // access private members of Node
};
```

# Ví dụ

```
class A {
    int a;
public:
    A() { a = 0; }
    friend class B;    // Friend Class
};

class B {
    int b;
public:
    void showA(A& x) {
        // Since B is friend of A, it can access private members of A
        std::cout << "A::a = " << x.a;
    }
};

int main() {
    A a;
    B b;
    b.showA(a);
}
```

A::a = 0

# Tóm tắt

---

- Khái niệm về đối tượng và lớp
- Khai báo và định nghĩa lớp
- Các thành phần của lớp
- Biến đối tượng và con trỏ đối tượng
- Con trỏ *this*
- Đối tượng là đối của hàm
- Hàm bạn và lớp bạn

# Bài tập

---

## 1. Xây dựng lớp Phân số (PS) có

- Các thành phần dữ liệu: ts, ms
- Các phương thức nhập(), xuất()
- Cải tiến xuất() để phân số in ra dưới dạng *tối giản*
- Hàm bạn để nhân hai phân số nhan()

## 2. Xây dựng lớp Số phức (SP) có:

- Các thành phần dữ liệu: a, b là phần thực và phần ảo
- Các phương thức nhập(), xuất() [dạng  $a+bi$ ]
- Phương thức cong() để cộng hai số phức

Xây dựng hàm main() trong đó sử dụng lớp SP

# Bài tập

---

## 3. Xây dựng lớp DT (Đa thức), trong đó:

- Các thuộc tính

int n;           // là bậc của đa thức

float \*a;       // là con trỏ xác định vùng bộ nhớ chứa các hệ số

- Phương thức nhap() để nhập các hệ số của đa thức
- Phương thức xuất() để in các hệ số của đa thức ra màn hình
- Phương thức gia\_tri(t) để tính giá trị của đa thức tại  $x = t$ .

[B3\_dathuc.cpp]



# Bài tập

---

## 4. Xây dựng lớp Điểm, gồm:

- Dữ liệu (private): hoành độ, tung độ
- Các phương thức (public):
  - nhap() để nhập dữ liệu
  - xuat() để xuất ra màn hình theo dạng (hoành độ, tung độ)
  - kc() để tính khoảng cách từ điểm hiện tại đến gốc tọa độ
  - kc(Diem d) để tính khoảng cách từ điểm hiện tại đến điểm d
- Hàm bạn (friend function)
  - kc(Diem d1, Diem d2) để tính khoảng cách giữa hai điểm d1 và d2
- Xây dựng hàm main() để sử dụng lớp trên.