

# Lập trình hướng đối tượng và C++

## Bài 4: Đối tượng và lớp

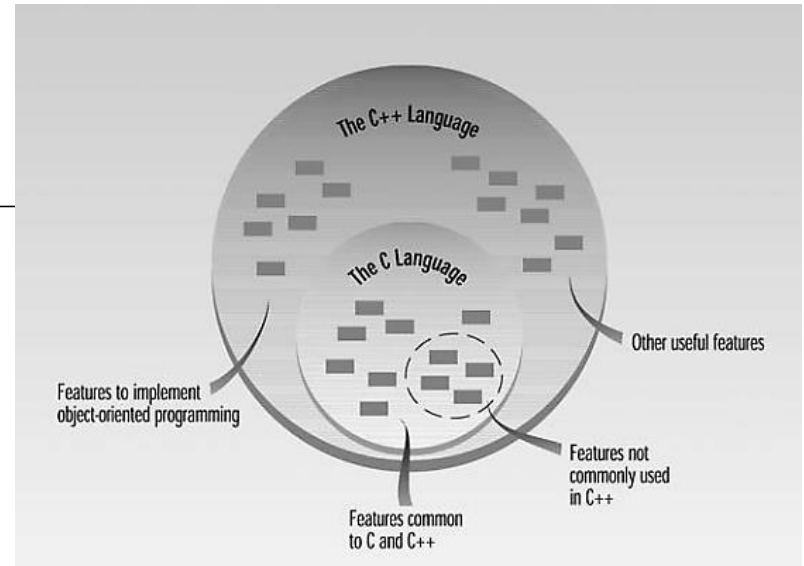
**TS. Nguyễn Hiếu Cường**

Bộ môn CNPM, Khoa CNTT, Trường Đại học GTVT

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

# Nội dung đã học

- Các kiểu lập trình (paradigms)
  - Hưởng chức năng (function-oriented)
  - Hưởng đối tượng (object-oriented)
- Hàm trong C++
  - Xây dựng hàm và sử dụng hàm (gọi hàm – calling function)
  - Tham số (đối): biến thông thường, con trỏ hoặc biến tham chiếu
  - Truyền tham số: call by value, call by pointer, call by reference
  - Đối có giá trị mặc định (default arguments)
  - Định nghĩa chồng hàm (function overloading)



# Nội dung chính

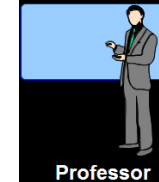
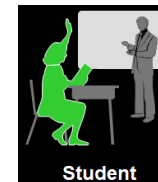
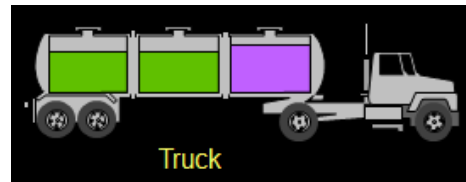
---

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

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

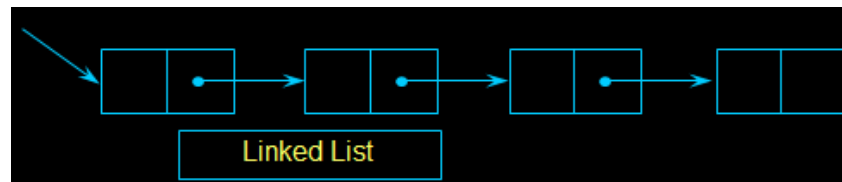
- Đối tượng vật lý



- 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?
- Liên hệ với các khái niệm đã biết
  - Lớp  $\approx$  Kiểu
  - Đối tượng  $\approx$  Biến

# Đối tượng và lớp

---

- Đối tượng  $\equiv$  Biến đối tượng
  - Các đối tượng được cấp bộ nhớ sau khi được tạo (khai báo hoặc new)
- Các đối tượng độc lập với nhau
  - Các đối tượng của cùng một lớp thì có các thuộc tính giống nhau
  - Nhưng giá trị các thuộc tính khác nhau ở các đối tượng
- Các đối tượng của cùng lớp sử dụng chung các hàm thành phần
  - 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

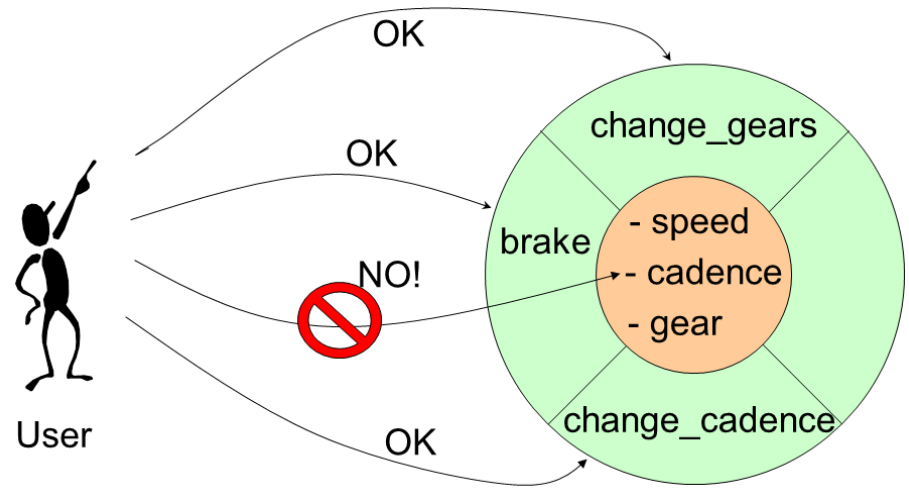
---

- Mỗi thành phần của lớp có phạm vi truy cập nhất định
- Phạm vi truy cập xác định bởi các từ khóa:
  - **private**  
Không thể truy nhập trực tiếp từ bên ngoài lớp, là chế độ mặc định
  - **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 d;  
    d.x = 50;  
    cout<< d.x;  
}
```

- Chương trình có lỗi, tại sao?

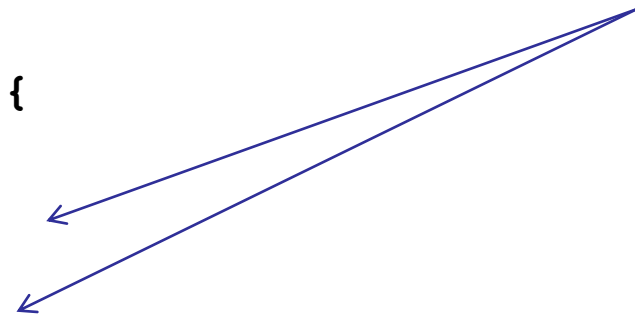
```
class A {  
  
    int x;  
};  
  
int main() {  
    A d;  
    d.x = 50;  
    cout<< d.x;  
}
```

# 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 d;  
    d.nhap();  
    d.xuat();  
}
```

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



# Biến đối tượng

---

- Khai báo biến đối tượng  
**Tên\_lớp Tên\_đối\_tượng ;**
- Truy nhập vào thành phần của đối tượng  
**Tên\_đối\_tượng . Tên\_thành\_phần**
- Truyền thông điệp cho một đối tượng  
**Tên\_đối\_tượng . Tên\_phương\_thức(...)**

Ví dụ:

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
- Khai báo con trỏ đối tượng

`Tên_lớp *Tên_con_trỏ;`

- Sử dụng con trỏ đối tượng

`Tên_con_trỏ -> Tên_thành_phần`

Ví dụ:

```
A  x, *p;  
x.nhap();      // Truy nhập vào thành phần qua đối tượng  
p = &x;  
p -> nhap();    // Truy nhập vào thành phần qua con trỏ đ/t
```

# Ví dụ

```
#include <iostream>
using namespace std;
// Khai báo lớp
class DIEM
{
    int x, y;
public:
    void nhapsl();
    void hien();
};

// Định nghĩa các phương thức
...
```

```
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();
}
```

# Ví dụ (lớp PS)

---

- Xây dựng lớp Phân số (PS) có
  - Các thành phần dữ liệu (private): ts, ms
  - Các phương thức (public): nhap(), xuất() và nhan()

Viết hàm main() để sử dụng lớp PS trên: nhập vào hai phân số, thực hiện nhân hai phân số đó, sau đó in phân số tích ra màn hình (không cần tối giản phân số).

# 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():

- Nhập tọa độ ba điểm
- In ra màn hình tọa độ ba điểm đó
- Tính chu vi và diện tích của tam giác có các đỉnh là ba điểm trên



# Con trỏ *this*

---

- C++ cung cấp một từ khóa *this* để thể hiện
  - Để trỏ đến *đối tượng hiện tại* (current object) và nó 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();
}
```

# Ví dụ

```
// Con trỏ this cần tường minh
#include <iostream>
#include <math.h>
using namespace std;
class DIEM
{
private:
    float x, y ;
public:
    void nhaps1() { ... }
    float kc(DIEM d2);
    float chuvi(DIEM d2, DIEM d3);
};
float DIEM::kc(DIEM d2)
{
    return(sqrt(...));
}
```

```
float DIEM::chuvi(DIEM d2, DIEM d3)
{
    float t= 0;
    t+= d2.kc(d3);
    t+= d3.kc(*this);
    t+= (*this).kc(d2);
    return t;
}
int main()
{
    DIEM d1, d2, d3;
    d1.nhaps1();
    d2.nhaps1();
    d3.nhaps1();
    cout<<"Chu vi tam giac = ";
    cout<<d1.chuvi(d2, d3);
}
```

# Bài tập (Xây dựng lớp)

---

Hãy xây dựng lớp Phân số (PS) có:

- Các thành phần dữ liệu (private): ts, ms
- Các phương thức (public):
  - nhap(),
  - xuất() [dạng ts/ms],
  - nhan() để nhân hai phân số (yêu cầu: phương thức có 1 đối),
  - cong() để cộng hai phân số (yêu cầu: phương thức có 2 đối).

Xây dựng hàm main() trong đó sử dụng lớp PS (yêu cầu: phân số in ra dưới dạng đã *tối giản*)

# 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

```
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
        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*
- Hàm bạn và lớp bạn

# Bài tập

---

## 1. Xây dựng lớp Số phức (SP) có:

- Các thành phần dữ liệu (private): phần thực và phần ảo
- Các phương thức (public): nhập(), xuất() và cong()
- Xây dựng hàm main() trong đó sử dụng lớp SP.

## 2. Xây dựng lớp Điểm có:

- Dữ liệu (private) gồm: hoành độ, tung độ
- Các phương thức (public):
  - Hàm nhập(), xuất() theo dạng (*hoành độ, tung độ*)
  - Hàm thành phần kc() tính khoảng cách từ một điểm đến gốc tọa độ
  - Hàm thành phần kc(Diem d) để tính khoảng cách giữa hai điểm
  - Hàm bạn kc(Diem d1, Diem d2) để tính khoảng cách giữa hai điểm d1 và d2
  - Xây dựng hàm main() để sử dụng lớp trên.

# Bài tập

---

## 3. 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():

- Nhập tọa độ ba điểm và tính diện tích của tam giác có các đỉnh là ba điểm trên.
- Nhập tọa độ của  $n$  điểm và tính độ dài đường gấp khúc lần lượt đi qua các điểm  $1, 2, 3, \dots, n-1, n$ .

# Bài tập

---

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