



LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Chương 4: Khởi tạo đối tượng, hàm bạn, lớp bạn (p2)



Trình bày: ThS. Lê Thanh Trọng



- 1. Giao diện và chi tiết cài đặt**
- 2. Hàm bạn**
- 3. Lớp bạn**
- 4. Các lưu ý khi xây dựng lớp**



- 1. Giao diện và chi tiết cài đặt**
2. Hàm bạn
3. Lớp bạn
4. Các lưu ý khi xây dựng lớp



Giao diện và chi tiết cài đặt

- ❖ Lớp có hai phần tách rời
 - Phần giao diện khai báo trong phần public để người sử dụng “thấy” và sử dụng
 - Chi tiết cài đặt bao gồm dữ liệu khai báo trong phần private của lớp và chi tiết mã hóa các hàm thành phần, vô hình đối với người dùng
- ❖ Lớp ThoiDiem có thể được cài đặt với các thành phần dữ liệu là giờ, phút, giây hoặc tổng số giây tính từ 0 giờ
- ❖ Khi thay đổi tổ chức dữ liệu của lớp cần đảm bảo không thay đổi phần giao diện thì không ảnh hưởng đến người sử dụng, và do đó không làm đổ vỡ kiến trúc của hệ thống



Lớp ThoiDiem – Cách 1

```
class ThoiDiem{  
    int Gio, Phut, Giay;  
    static bool KiemTraHopLe(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 GetGio() const {return Gio; }  
    int GetPhut() const {return Phut; }  
    int GetGiay() const {return Giay; }  
    void Nhap();  
    void Xuat() const;  
    void Tang();  
    void Giam();  
};
```



Lớp ThoiDiem – Cách 2

```
class ThoiDiem{  
    long TongSoGiay;  
    static bool KiemTraHopLe(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 GetGio() const {return TongSoGiay/3600;}  
    int GetPhut() const {return (TongSoGiay %3600)/60;}  
    int GetGiay() const {return TongSoGiay %60;}  
    void Nhap();  
    void Xuat() const;  
    void Tang();  
    void Giam();  
};
```



1. Giao diện và chi tiết cài đặt
2. Hàm bạn
3. Lớp bạn
4. Các lưu ý khi xây dựng lớp



Hàm bạn, lớp bạn

- ❖ Giả sử có lớp Vector, lớp Matrix
- ❖ Cần viết hàm nhân Vector với một Matrix

❖ Hàm nhân:

- Không thể thuộc lớp Vector
- Không thể thuộc lớp Matrix
- Không thể tự do

→ Giải pháp:

- Xây dựng hàm truy cập dữ liệu?
- Sử dụng hàm bạn/ lớp bạn



Hàm bạn (Friend function)

- ❖ Hàm bạn không thuộc lớp, không phụ thuộc vào lớp
- ❖ Có quyền truy cập tất cả các thành phần của lớp (kể cả private)
- ❖ Cho phép chia sẻ dữ liệu giữa các đối tượng với một hàm tùy ý trong chương trình (hàm friend) hoặc chia sẻ các thành phần của đối tượng có thuộc tính private hay protected với các đối tượng khác (lớp friend)



Hàm bạn (Friend function)

- ❖ Một lớp có thể khai báo một hay nhiều hàm “bạn”
- ❖ Ưu điểm: Kiểm soát các truy nhập ở cấp độ lớp – không thể áp đặt hàm bạn cho lớp nếu điều đó không được dự trù trước trong khai báo của lớp
- ❖ Dùng từ khóa **friend** để khai báo, định nghĩa hàm bạn
- ❖ Có thể định nghĩa hàm bạn ở trong hoặc ngoài lớp
- ❖ Nếu định nghĩa bên ngoài thì không cần khai báo lại từ khóa **friend**



Hàm bạn (Friend function)

❖ Các tính chất của quan hệ friend:

- ❑ Phải được cho, không được nhận
- ❑ Không đối xứng, không bắc cầu

❖ Quan hệ friend có vẻ như vi phạm khái niệm đóng gói (encapsulation) của OOP nhưng có khi lại cần đến nó để cài đặt các mối quan hệ giữa các lớp và khả năng đa năng hóa toán tử trên lớp (sẽ đề cập ở chương sau)



Ví dụ hàm bạn

```
class CounterClass{  
    int Counter;  
public:  
    char CounterChar;  
    void Init( char );  
    void AddOne( ){  
        Counter++;  
    }  
    friend int Total (int);  
};
```



Ví dụ hàm bạn

```
CounterClass MyCounter[26];           //Có 26 đối tượng  
  
int Total(int NumberObjects)  
{  
    for (int i=0, sum=0; i<NumberObjects; i++)  
        sum += MyCounter[i].Counter;  
    //Tính tổng số ký tự trong số các Objects ký tự  
    return sum;  
}
```



Hàm bạn (Friend function)

❖ Lưu ý:

- ❑ Vị trí của khai báo “bạn bè” trong lớp hoàn toàn tùy ý
- ❑ Trong hàm bạn, không còn tham số ngầm định this như trong hàm thành phần
- ❑ Hàm bạn của một lớp có thể có một hay nhiều tham số, hoặc có thể có giá trị trả về thuộc kiểu lớp đó



1. Giao diện và chi tiết cài đặt
2. Hàm bạn
- 3. Lớp bạn**
4. Các lưu ý khi xây dựng lớp



Lớp bạn (Friend class)

- ❖ Một lớp (A) có thể truy cập đến các thành phần có thuộc tính private của một lớp khác (B), khi đó A được khai báo là bạn của B
- ❖ Để thực hiện được điều này, chúng ta có thể lấy toàn bộ một lớp là bạn (lớp friend) của lớp khác

```
class A
{
    //các khai báo
};

class B
{
    //các khai báo
friend class A; //A là bạn của B
};
```



Ví dụ lớp bạn

```
class Tom{  
public:  
    friend class Jerry;          //Có lớp bạn là Jerry  
private:  
    int SecretTom;             //Bí mật của Tom  
};  
  
class Jerry{  
public:  
    void Change(Tom T){  
        T.SecretTom++;          //Bạn nên có thể truy cập  
    }  
};
```



- 1. Giao diện và chi tiết cài đặt**
- 2. Hàm bạn**
- 3. Lớp bạn**
- 4. Các lưu ý khi xây dựng lớp**

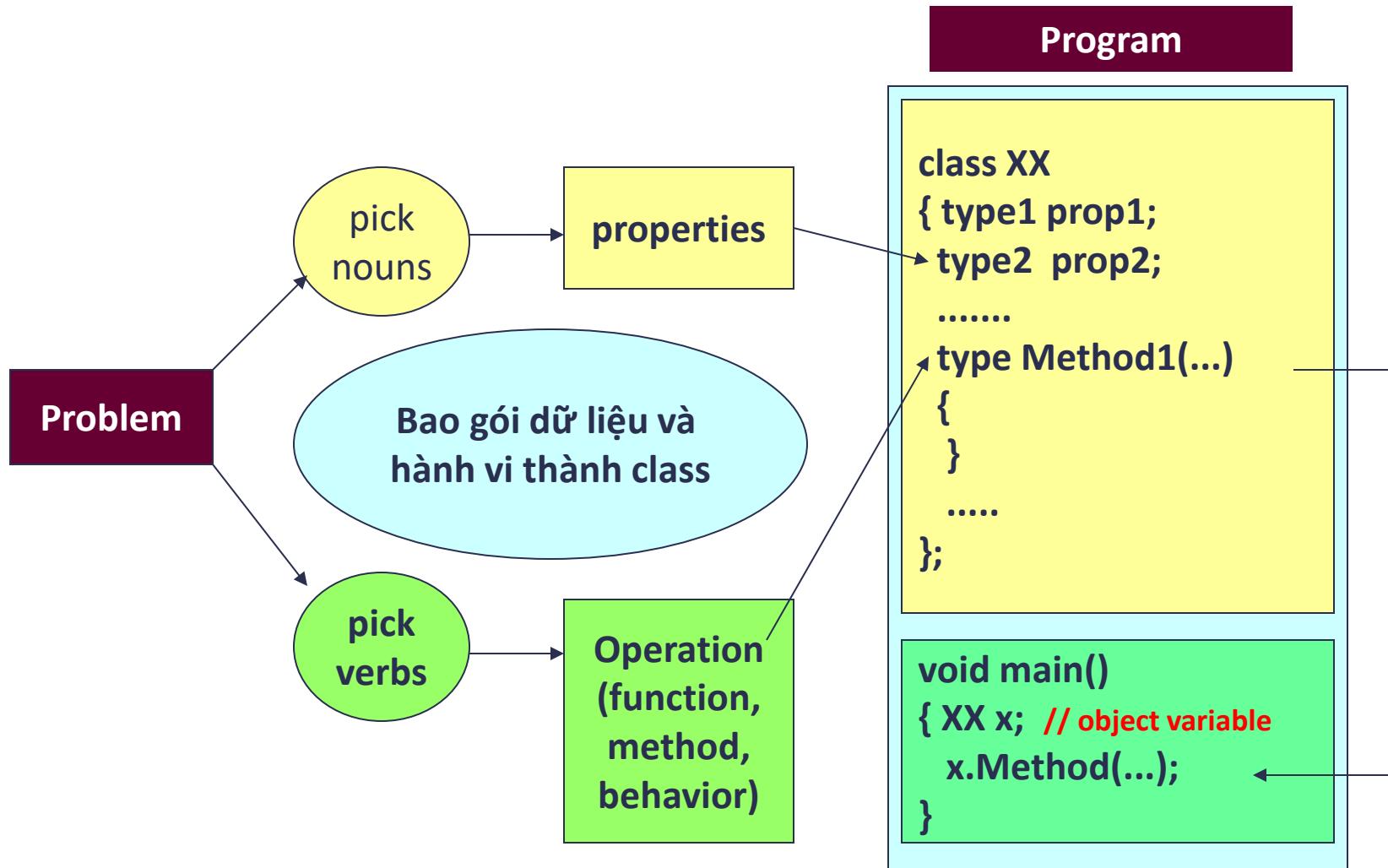


Các lưu ý khi xây dựng lớp

- ❖ Hình thành lớp: Khi ta nghĩ đến “nó” như một khái niệm riêng lẻ
→ Xây dựng lớp biểu diễn khái niệm đó
- ❖ Lớp là biểu diễn cụ thể của một khái niệm vì vậy tên lớp luôn là **danh từ**
- ❖ Các thuộc tính của lớp là các thành phần dữ liệu nên chúng luôn là **danh từ**
- ❖ Các hàm thành phần (các hành vi) là các thao tác chỉ rõ hoạt động của lớp nên các hàm là **động từ**



Các lưu ý khi xây dựng lớp





Các lưu ý khi xây dựng lớp

- ❖ Các thuộc tính có thể suy diễn từ những thuộc tính khác thì dùng hàm thành phần để thực hiện tính toán
- ❖ Ví dụ chu vi, diện tích của một tam giác

```
class TamGiac{  
    Diem A,B,C;  
    double ChuVi;  
    double DienTich;  
public:  
    //...  
};
```

```
class TamGiac{  
    Diem A,B,C;  
public:  
    //...  
    double ChuVi() const;  
    double DienTich() const;  
};
```



Các lưu ý khi xây dựng lớp

- ❖ Tuy nhiên, nếu các thuộc tính suy diễn đòi hỏi nhiều tài nguyên hoặc thời gian để thực hiện tính toán, ta có thể khai báo là dữ liệu thành phần

```
class QuocGia{  
    long DanSo;  
    double DienTich;  
    double TuoiTrungBinh;  
public:  
    double TinhTuoiTB() const;  
    //...  
};
```



Các lưu ý khi xây dựng lớp

- ❖ Dữ liệu thành phần nên được kết hợp:

```
class TamGiac{
```

```
    Diem A,B,C;
```

```
public:
```

```
    //...
```

```
};
```

```
class HinhTron{
```

```
    Diem Tam;
```

```
    double BanKinh;
```

```
public:
```

```
    //...
```

```
};
```

```
class TamGiac{
```

```
    double xA, yA;
```

```
    double xB, yB, xC, yC;
```

```
public:
```

```
    //...
```

```
};
```

```
class HinhTron{
```

```
    double tx, ty, BanKinh;
```

```
public:
```

```
    //...
```

```
};
```



Các lưu ý khi xây dựng lớp

- ❖ Trong mọi trường hợp, nên có phương thức thiết lập (constructor) để khởi động đối tượng
- ❖ Nên có phương thức thiết lập có khả năng tự khởi động không cần tham số
- ❖ Nếu đối tượng có nhu cầu cấp phát tài nguyên thì phải có phương thức thiết lập, copy constructor để khởi động đối tượng bằng đối tượng cùng kiểu và có destructor để dọn dẹp. Ngoài ra còn có phép gán (chương 4)
- ❖ Nếu đối tượng đơn giản không cần tài nguyên riêng → Không cần copy constructor và destructor



Tóm tắt về hàm bạn, lớp bạn và quy tắc xây dựng lớp

❖ Lưu ý khi chỉnh sửa lớp

- ❑ Lớp có hai phần tách rời: phần giao diện ([các phương thức](#)) khai báo trong phần public để người sử dụng “thấy” và sử dụng và chi tiết cài đặt bao gồm dữ liệu khai báo trong phần private của lớp và chi tiết mã hóa các hàm thành phần, vô hình đối với người dùng
- ❑ Khi thay đổi tổ chức dữ liệu của lớp cần **đảm bảo không thay đổi phần giao diện** thì không ảnh hưởng đến người sử dụng, và do đó không làm đổ vỡ kiến trúc của hệ thống



Tóm tắt về hàm bạn, lớp bạn và quy tắc xây dựng lớp

❖ Hàm bạn/ Lớp bạn

❑ **friend** là một từ khóa được sử dụng để khai báo hàm hoặc lớp có quyền truy cập vào các thành viên riêng tư (private) và được bảo vệ (protected) của một lớp khác

❑ Việc sử dụng friend giúp thực hiện các thao tác phức tạp mà không cần lạm dụng việc mở rộng phạm vi truy cập của các thành viên lớp đó

❑ Hàm bạn

- Là một hàm không phải là thành viên của lớp, nhưng có quyền truy cập vào các thành viên riêng tư (private) và được bảo vệ (protected) của lớp đó
- Cú pháp: sử dụng từ khóa **friend** bên trong định nghĩa của lớp
- Ví dụ: **friend void DisplayData(MyClass& obj); //hàm bạn của lớp MyClass**

❑ Lớp bạn

- Là một lớp mà tất cả các hàm thành viên của nó có quyền truy cập vào các thành viên riêng tư (private) và được bảo vệ (protected) của lớp khác
- Cú pháp: sử dụng từ khóa **friend** bên trong định nghĩa của lớp
- Ví dụ: **class ClassA { friend class ClassB; };**



Tóm tắt về hàm bạn, lớp bạn và quy tắc xây dựng lớp

❖ Các lưu ý khi xây dựng lớp

- ❑ Nguyên tắc đóng gói (Encapsulation)
- ❑ Nguyên tắc trừu tượng hóa (Abstraction)
- ❑ Nguyên tắc kế thừa (Inheritance)
- ❑ Nguyên tắc đa hình (Polymorphism)
- ❑ Nguyên tắc đơn trách nhiệm (Single Responsibility Principle)
- ❑ Nguyên tắc đóng mở (Open/Closed Principle)
- ❑ ...



Tóm tắt về hàm bạn, lớp bạn và quy tắc xây dựng lớp

❖ Các lưu ý khi xây dựng lớp

- ❑ Tên lớp luôn là danh từ
- ❑ Các thuộc tính của lớp luôn là danh từ
- ❑ Các hàm thành phần (các hành vi) nên là động từ
- ❑ Các thuộc tính có thể suy diễn từ những thuộc tính khác thì dùng hàm thành phần để thực hiện tính toán. Tuy nhiên, nếu các thuộc tính suy diễn đòi hỏi nhiều tài nguyên hoặc thời gian để thực hiện tính toán, ta có thể khai báo là dữ liệu thành phần
- ❑ Trong mọi trường hợp, nên có phương thức thiết lập (constructor) để khởi động đối tượng
- ❑ Nên có phương thức thiết lập có khả năng tự khởi động không cần tham số
- ❑ Nếu đối tượng có nhu cầu cấp phát tài nguyên thì phải có phương thức thiết lập, copy constructor để khởi động đối tượng bằng đối tượng cùng kiểu và có destructor để dọn dẹp
- ❑ Nếu đối tượng đơn giản (không chứa con trỏ hay đối tượng phức tạp) thì không cần copy constructor và destructor