

Lập trình hướng đối tượng

CHƯƠNG 2

XÂY DỰNG VÀ SỬ DỤNG LỚP

Nội dung

1. Xác định các lớp đối tượng, các thuộc tính và các phương thức, thiết kế sơ đồ lớp.
2. Đọc sơ đồ lớp, cài đặt lớp.
3. Phạm vi truy xuất
4. Sử dụng lớp.
5. Cài đặt chương trình ứng dụng hướng đối tượng
6. Phương thức getter(), setter()
7. Quan hệ kết tập (lồng nhau)
8. Mảng đối tượng
9. Hàm bạn, lớp bạn.
10. Con trỏ đối tượng
11. Phương thức khởi tạo

2.1. Xác định các lớp đối tượng

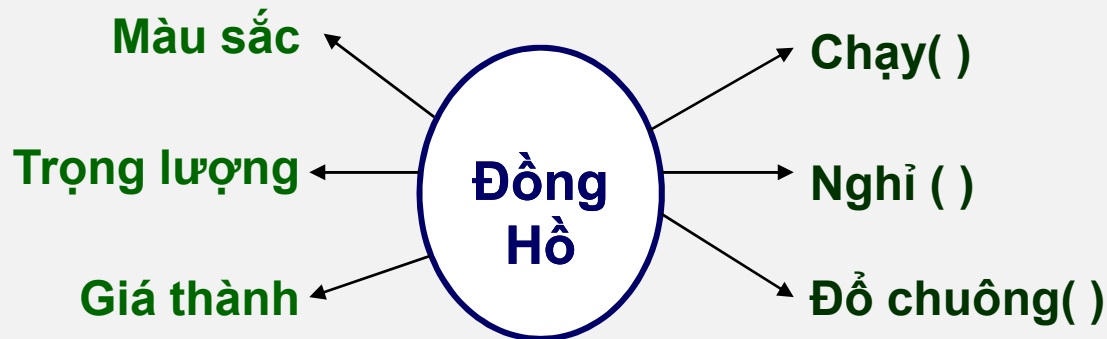
- Tìm các danh từ và động từ
 - Danh từ chung -> các lớp đối tượng
 - Danh từ riêng -> các đối tượng
- Xác định các lớp cần thiết
- Xác định các phương thức <- động từ
- Xác định các thuộc tính <- danh từ
- Xác định quan hệ kết tập
- Xác định quan hệ kế thừa

Xác định các lớp đối tượng (tt)

- Ví dụ: Mô phỏng quần thể đồng hồ

Động từ	Danh từ	
Chạy	Đồng hồ	Màu sắc
Nghỉ		Trọng lượng
Đổ chuông		Giá thành

Sơ đồ lớp



Đồng Hồ
Màu sắc
Trọng lượng
Giá thành
Chạy()
Nghỉ()
Đổ chuông()

2.2. Định nghĩa lớp

- **Mẫu 1:**
 - Các thuộc tính và phương thức được định nghĩa trong thân lớp.

```
class <Tên_Lớp>
{
    // Định nghĩa các thuộc tính
    // Định nghĩa các phương thức
};
```

Định nghĩa lớp (tt)

- **Mẫu 2:**

- Các thuộc tính và nguyên mẫu phương thức được định nghĩa trong thân lớp.
- Nội dung phương thức được định nghĩa bên ngoài lớp.

```
class <Tên_Lớp>
{
    // Định nghĩa các thuộc tính
    // Khai báo nguyên mẫu các phương thức
};
//Định nghĩa các phương thức
```

Định nghĩa lớp (tt)

- Giải thích

- <Tên_Lớp>: do người dùng đặt tùy ý, viết theo đúng quy tắc đặt tên của ngôn ngữ lập trình C++.
- Định nghĩa các thuộc tính:

```
<kiểu_dữ_liệu> <tên_thuộc_tính>;
```

- Khai báo và định nghĩa phương thức trong thân lớp

```
<kiểu_trả_về> <tên_PT>([danh sách đối])  
{  
    // Thân phương thức;  
}
```

Định nghĩa lớp (tt)

- Định nghĩa phương thức bên ngoài thân lớp

```
<kiểu_trả_về> <Tên_Lớp>::<tên_PT>([ds đối])  
{  
    //Nội dung phương thức  
}
```

:: gọi là toán tử phạm vi

Định nghĩa lớp (tt) – Ví dụ mẫu 1

```
class Nguoi{
    private:
        char hoTen[30];
        int tuoi;
        char que[60];
    public:
        void nhap()
        { cout<<"Nhap ho ten ";
          fflush(stdin); gets(hoTen);
          cout<<"Nhap tuoi"; cin>>tuoi;
          cout<<"Nhap que quan ";
          fflush(stdin); gets(que);
        }
        void xuat() {
            cout<<"Ho va ten:"<<hoTen<<endl;
            cout<<"Tuoi "<<tuoi<<endl;
            cout<<"Que quan: "<<que<<endl;
        }
};
```

Định nghĩa lớp (tt) – Ví dụ mẫu 2

```
class Nguoi{
    private:
        char hoTen[30];
        int tuoi;
        char que[45];
    public:
        void nhap();
        void xuat();
};

void Nguoi::nhap() {
    cout<<"Nhap ho ten "; flush(stdin);gets(hoTen);
    cout<<"Nhap tuoi"; cin>>tuoi;
    cout<<"Nhap que quan ";fflush(stdin);gets(que);
}

void Nguoi::xuat() {
    cout<<"Ho va ten: "<<hoTen<<endl;
    cout<<"Tuoi: "<<tuoi<<endl;
    cout<<"Que quan: "<<que<<endl;
}
```

2.3. Phạm vi truy xuất

- Phạm vi truy xuất được thiết lập cho các thành phần của lớp gồm các thuộc tính và các phương thức.
- Phạm vi truy xuất xác định phần mã lệnh nào của chương trình được phép truy xuất vào các thành phần của lớp.
- Các loại phạm vi truy xuất gồm:
 - **private** – riêng tư
 - **protected** – bảo vệ
 - **public** – công cộng

Phạm vi truy xuất (tt)

- **private:** Các thành phần có phạm vi **private** chỉ được truy xuất bởi các phương thức trong nội bộ lớp.
- **protected:** Các thành phần có phạm vi **protected** có thể được truy xuất bởi các phương thức bên trong lớp và các phương thức của lớp dẫn xuất trực tiếp từ lớp có các thành phần này.
- **public:** Các thành phần có phạm vi **public** có thể được truy xuất bởi mọi mã lệnh trong chương trình.

Phạm vi truy xuất (tt)

- Cách đặt từ khóa chỉ định phạm vi truy xuất:

<từ_khóa_phạm_vi:>

- Vị trí đặt từ khóa chỉ định phạm vi truy xuất:
 - Trước các thành phần cần xác định phạm vi truy xuất.
- Lưu ý:
 - Nếu không đặt từ khóa chỉ định phạm vi truy xuất trước các thành phần của lớp thì mặc định các thành phần này có phạm vi truy xuất là **private**.

Phạm vi truy xuất (tt) – Ví dụ

```
class Nguoi
{
    private:
        char hoTen[30];
        char ngaySinh[11];
        float chieuCao;
    public:
        void nhap();
        void xuat();
};
```

Phạm vi truy xuất (tt) – Ví dụ

```
class Nguoi
{
    char hoTen[30];
    char ngaySinh[11];
    float chieuCao;

    public:
        void nhap();
        void xuat();
};
```

Các thành phần này có
phạm vi truy xuất là
private

2.4. Sử dụng lớp

- Chương trình thực thi do sự hoạt động của các đối tượng được tạo ra trong chương trình.
- Trong chương trình, các lớp được xem như các kiểu dữ liệu, các đối tượng là các biến.
- Việc sử dụng lớp trong chương trình để khai báo các đối tượng cũng giống như sử dụng các kiểu dữ liệu để khai báo các biến.
- Để tạo ra các đối tượng trong chương trình cần khai báo thông qua các lớp đã được định nghĩa.

Sử dụng lớp (tt)

- Khai báo đối tượng

`<Tên_lớp> <tên_đối_tượng>;`

- Ví dụ

`Ngnoi nguoi_1, nguoi_2, nguoi[20];`

- Trong khai báo trên:

`nguoi_1, nguoi_2`: là các đối tượng thuộc lớp `Ngnoi`.

`nguoi[20]`: là mảng chứa 20 đối tượng thuộc lớp `Ngnoi`.

Sử dụng lớp (tt)

- Truy xuất phương thức của đối tượng

`<tên_đối_tượng>.<tên_phương_thức>`

- Ví dụ

`nguo_i_1.nhap();`

`nguo_i_1.xuat();`

- Truy xuất vào các thuộc tính của đối tượng

`<tên_đối_tượng>.<thuộc_tính>`

- Ví dụ

`nguo_i_1.hoTen, nguo_i[2].ngaySinh`

2.5. Cài đặt chương trình ứng dụng

1. Viết chương trình hướng đối tượng giải quyết bài toán tính diện tích hình thang.
2. Viết chương trình hướng đối tượng giải quyết bài toán tính chu vi, diện tích hình tròn.
3. Viết chương trình hướng đối tượng giải quyết bài toán giải phương trình bậc nhất một ẩn.
4. Viết chương trình hướng đối tượng giải quyết bài toán giải phương trình bậc 2.
5. Viết chương trình hướng đối tượng giải quyết bài toán tính chu vi và diện tích của một tam giác.
6. Viết chương trình hướng đối tượng giải quyết bài toán, nhập và hiển thị thông tin của một học sinh gồm: Họ tên, ngày, tháng, năm sinh, giới tính, điểm trung bình, xếp loại đạo đức.

Cài đặt chương trình ứng dụng (tt)

7. Cài đặt chương trình hướng đối tượng thực hiện các yêu cầu:

- Thiết kế, cài đặt lớp CanBo bao gồm các thuộc tính: Mã cán bộ, họ và tên, ngày sinh, số ngày làm việc trong tháng và các phương thức:
- Phương thức nhap() để nhập thông tin của cán bộ.
- Phương thức tinhLuong(): Trả về lương của cán bộ theo công thức $Lương = Số\ ngày\ làm\ việc\ trong\ tháng * 250.000$.
- Phương thức xuất(): hiển thị thông tin của cán bộ ra màn hình.
- Xây dựng chương trình chính nhập thông tin cho 1 cán bộ. Xuất thông tin của cán bộ đó ra màn hình và cho biết Lương của cán bộ đó là bao nhiêu.

2.6. Phương thức getter(), setter()

- Phương thức getter(): Để trả về giá trị thuộc tính của đối tượng.


- Ví dụ:

```
class Nguoi{  
    private:  
        char hoTen[30];  
        char ngaySinh[11];  
        float chieuCao;  
    public:  
        ...  
        float getChieuCao() {  
            return chieuCao;  
        }  
};
```

Phương thức getter(), setter()

- Phương thức getter(): Sử dụng khi muốn lấy giá trị của thuộc tính **private** của đối tượng (từ bên ngoài lớp).

```
class Ngnoi{  
    ...  
    public:  
        ...  
        float getChieuCao() {  
            return chieuCao;  
        }  
};  
  
int main() {  
    Ngnoi n;  
    cout<<"Chieu cao la: "<<n.getChieuCao();  
}
```



Phương thức getter(), setter()

- Phương thức setter(): Để thiết lập giá trị mới cho thuộc tính của đối tượng.

- Ví dụ:

```
class Nguoi{  
    private:  
        char hoTen[30];  
        char ngaySinh[11];  
        float chieuCao;  
    public:  
        ...  
        void setChieuCao(float t) {  
            chieuCao = t;  
        }  
};
```

Phương thức getter(), setter()

- Phương thức setter(): Sử dụng khi muốn thiết lập giá trị mới cho thuộc tính **private** của đối tượng (từ bên ngoài lớp).

```
class Ngnoi{  
    ...  
    public:  
        ...  
        void setChieuCao(float t) {  
            chieuCao = t;  
        }  
};  
int main() {  
    Ngnoi n;  
    n.setChieuCao(1.75); //n.chieuCao = 1.75  
}
```


Phương thức getter(), setter() – Ví dụ

- **Chương trình giải quyết bài toán:**

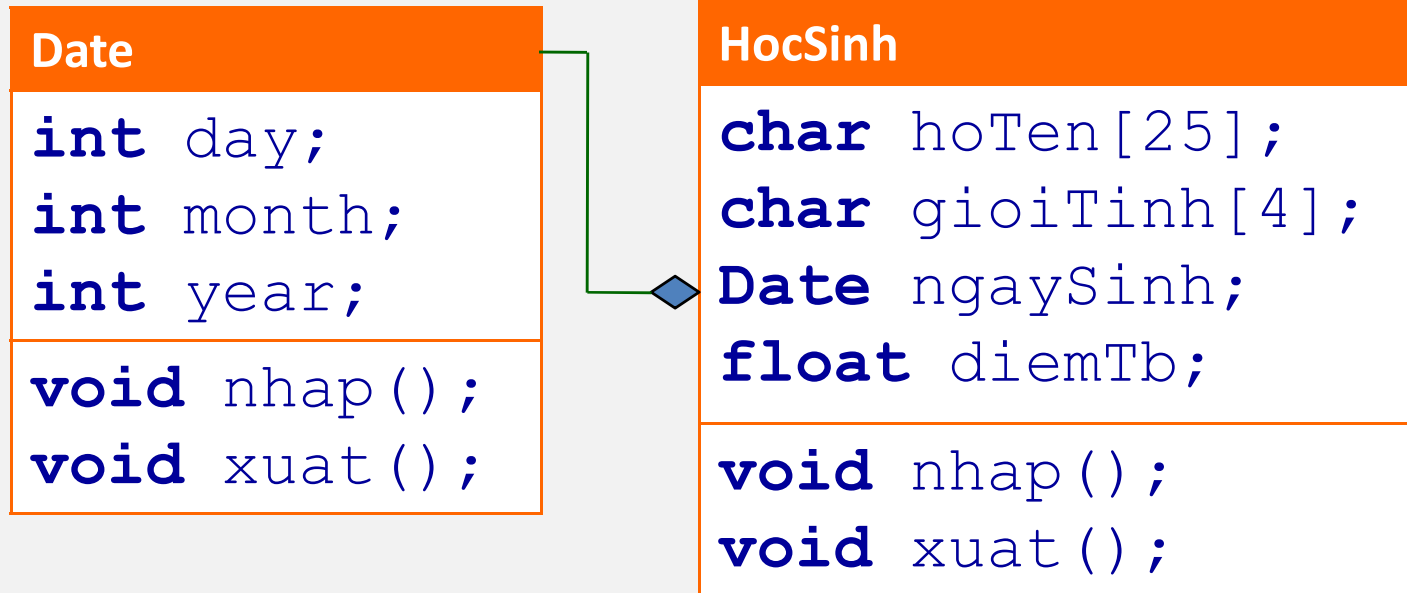
Tạo ra một đối tượng học sinh và khởi tạo các thông tin cho đối tượng gồm: Họ tên, ngày, tháng, năm sinh, giới tính, điểm trung bình, xếp loại đạo đức.

Hiển thị thông tin của học sinh ra màn hình.

- **Chương trình sử dụng phương thức setter(), getter().**

2.7. Quan hệ kết tập

- Khi thuộc tính của một lớp là một đối tượng của lớp khác.
- Sơ đồ lớp cho quan hệ kết tập.



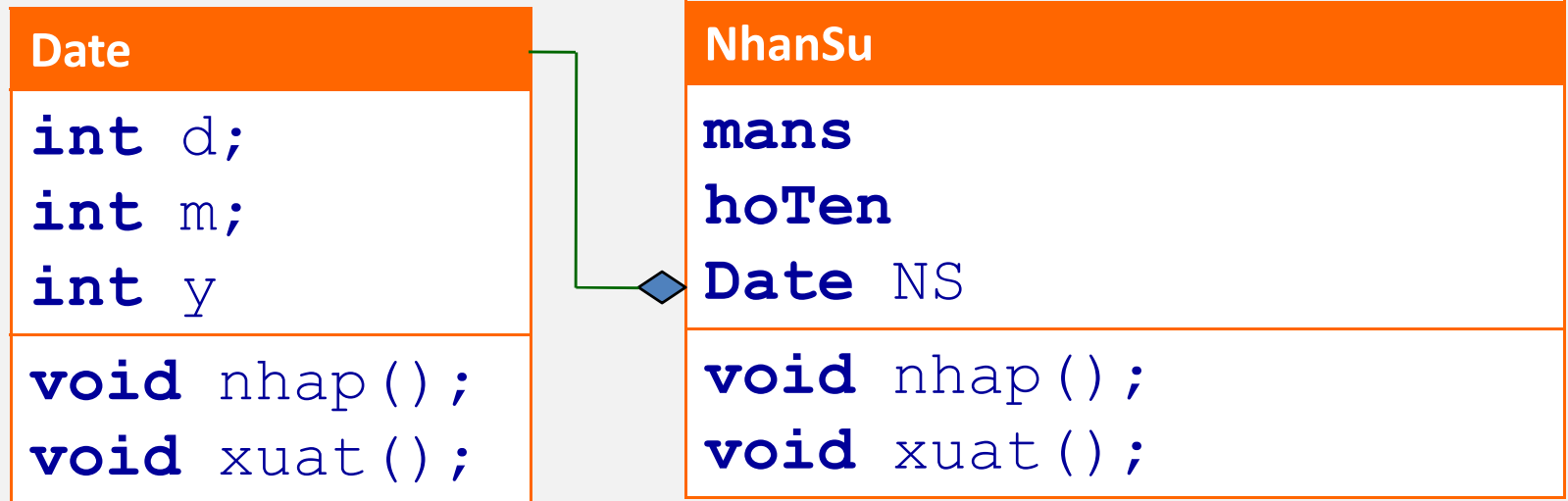
Quan hệ kết tập (tt)

```
class Date {  
    private:  
        int day, month, year;  
    ...  
};  
class HocSinh {  
    private:  
        char hoTen[30];  
        char gioiTinh[4];  
        Date ngaySinh;  
        float diemTb;  
    public:  
        ...  
};
```

- Thuộc tính ngaySinh của lớp HocSinh là một đối tượng của lớp Date.
- Cần chú ý việc truy xuất vào các thuộc tính **private** của lớp Date.

Quan hệ kết tập (tt) – Ví dụ 1

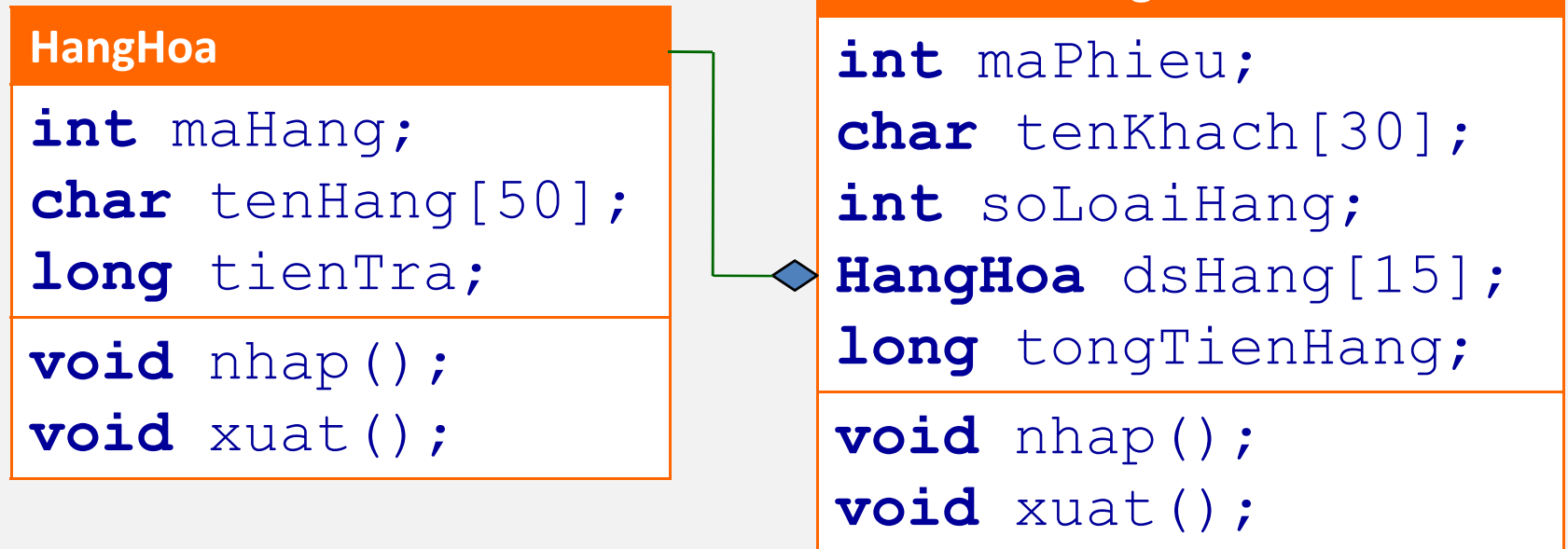
- Cài đặt chương trình theo sơ đồ lớp dưới đây



- Yêu cầu: Cài đặt hàm main nhập vào một nhân sự x, in thông tin của nhân sự ra màn hình.

Quan hệ kết tập (tt) – Ví dụ 2

- Cài đặt chương trình theo sơ đồ lớp dưới đây



- Yêu cầu: Nhập thông tin cho một phiếu mua hàng, hiển thị thông tin phiếu mua hàng ra màn hình.

2.8. Hàm bạn – Lớp bạn

- Hàm bạn của một lớp
 - Để truy xuất vào các thành phần riêng tư của lớp
- Khai báo hàm bạn

```
class A{  
    private:  
        //Khai báo các thuộc tính của lớp A  
    public:  
        //Khai báo các phương thức của lớp A  
    //Khai báo hai hàm bạn f1 và f2  
    friend <nguyên mẫu của hàm f1>;  
    friend <nguyên mẫu của hàm f2>;  
};  
  
// Xây dựng các hàm f1 và f2 bên ngoài lớp A
```

Tính chất – đặc điểm của hàm bạn

- Hàm không nằm trong phạm vi của lớp mà nó đã được khai báo là friend.
- Nó không thể được gọi bằng cách sử dụng đối tượng vì nó không nằm trong phạm vi của lớp đó.
- Nó có thể được gọi như một hàm bình thường mà không cần sử dụng đối tượng.
- Nó không thể truy cập trực tiếp vào tên thành viên và phải sử dụng tên đối tượng và dấu chấm toán tử với tên thành viên.
- Nó có thể được khai báo trong phần private hoặc public.
- (Trong thân hàm bạn của một lớp có thể truy nhập tới các thuộc tính của các đối tượng thuộc lớp này. Đây là khác nhau duy nhất giữa hàm bạn và hàm thông thường.
 - + Hàm bạn không phải là phương thức của một lớp, lời gọi của hàm bạn giống như lời gọi của hàm thông thường.
-)

2.7.1. Hàm bạn – Ví dụ

```
class A{
    private:
        int a, b;
    public:
        void set(){
            a = 2; b = 3;
        }
    // Khai báo hàm tinh() là bạn của A
        friend void tinh(A dt);
};
void tinh(A dt){
    cout<<"Dien tich: "<<dt.a*dt.b;
}
int main(){
    A a;
    a.set();  tinh(a);
}
```


2.7.2. Lớp bạn

Khi tất cả các phương thức của một lớp là bạn của một lớp khác, thì lớp của các phương thức đó cũng trở thành lớp bạn của lớp kia. Muốn khai báo một lớp B là lớp bạn của lớp A.

Lớp friend được xây dựng để khắc phục điểm yếu **lớp dẫn xuất không thể truy cập tới các biến private của lớp cơ sở**

a. Định nghĩa:

Một friend có thể là một hàm, một mẫu hàm, hoặc hàm thành viên, hoặc một lớp hoặc một mẫu lớp, trong trường hợp này, toàn bộ lớp và tất cả thành viên của nó là friend.

Hàm friend trong C++ của một lớp được định nghĩa bên ngoài phạm vi lớp đó, nhưng nó có quyền truy cập tất cả thành viên private và protected của lớp đó. Ngay cả khi các nguyên mẫu cho hàm friend xuất hiện trong định nghĩa lớp, thì các hàm friend không là các hàm thành viên.

2.7.2. Lớp bạn

b.Tính chất:

- Friend của một class có thể là thành viên của 1 class khác
- Friend của 1 class có thể là thành viên của class khác hoặc tất cả các class trong cùng 1 chương trình. Như là 1 GLOBAL FRIEND
- Friend có thể access private hoặc protected của class được khai báo là Friend.
- Friends không phải là một thành viên vì vậy không có con trỏ "this"
- Friend có thể khai báo ở bất cứ đâu (public, protected or private section) trong một class.

2.7.2. Lớp bạn

c.Khai báo tuần tự theo sau:

Khai báo khuôn mẫu lớp B:

class B;

Định nghĩa lớp A, với khai báo B là lớp bạn:

class A

{

...// Khai báo các thành phần của lớp A

// Khai báo lớp bạn B

friend class B;

};

Định nghĩa chi tiết lớp B:

class B

{ ... // Khai báo các thành phần của lớp B };

Lưu ý:

- Trong trường hợp này, lớp B là lớp bạn của lớp A nhưng không có nghĩa là lớp A cũng là bạn của lớp B: tất cả các phương thức của lớp B có thể truy cập các thành phần private của lớp A (thông qua các đối tượng có kiểu lớp A) nhưng các phương thức của lớp A lại không thể truy cập đến các thành phần private của lớp B.
- Muốn các phương thức của lớp A cũng truy cập được đến các thành phần private của lớp B, thì phải khai báo thêm là lớp A cũng là lớp bạn của lớp B.

2.7.2. Lớp bạn

- Lớp bạn của một lớp: Để các phương thức của lớp có thể truy xuất vào các thành phần riêng tư của lớp khác.
- Ví dụ:

```
class A{  
    private:  
        int a, b;  
    public:  
        void nhap() {  
            cout<<"Nhap a, b: ";  
            cin>>a>>b;  
        }  
};
```

```
class B{  
    private:  
        int c;  
    public:  
        void tinh(A dt) {  
            cout<<"dien tich:";  
            cout<<dt.a * dt.b;  
        }  
};
```

- Trong trường hợp này chương trình báo lỗi vì **tinh()** không phải là phương thức của lớp A.

Lớp bạn (tt)

- Khi đó lớp B phải là bạn của lớp A

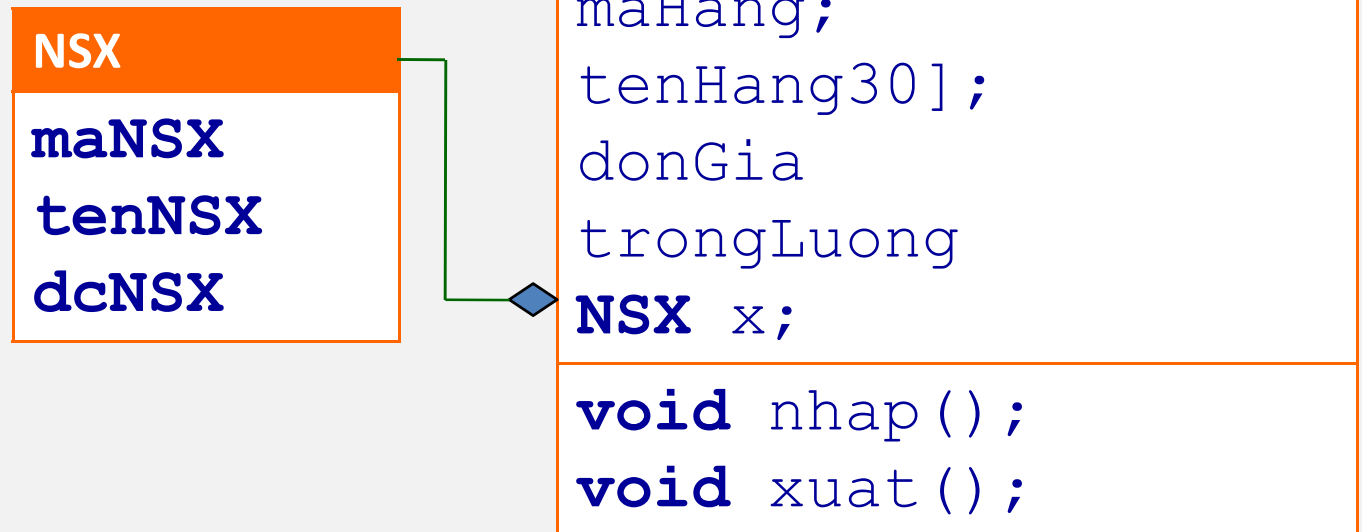
```
class A {  
    private:  
        int a, b;  
    public:  
        void nhap() {  
            cout<<"Nhap a, b ";  
            cin>>a>>b;  
        }  
    //KB B la ban cua lop A  
    friend class B;  
};
```

```
class B{  
    public:  
        void tinh(A dt) {  
            cout<<"D.tich: ";  
            cout<< dt.a * dt.b;  
        }  
};  
int main() {  
    A d1;  
    B d2;  
    d1.nhap();  
    d2.tinh(d1);  
}
```

Chương trình này chạy tốt!

Lớp bạn (tt) – Ví dụ

- Cài đặt chương trình theo sơ đồ lớp dưới đây



- Yêu cầu: Nhập thông tin của một mặt hàng, hiển thị thông tin của mặt hàng đó ra màn hình.

2.9. Mảng đối tượng

- Mảng đối tượng được tạo ra để lưu trữ danh sách đối tượng trong chương trình.
- Khai báo mảng đối tượng.

```
<Tên_Lớp> <tên_mảng><[kích_thước]>;
```

- Mảng đối tượng cũng giống các mảng khác, với mỗi phần tử mảng là (dùng để chứa) một đối tượng.
- Ví dụ:

```
CanBo cb[20];
```

```
//Mảng chứa 20 đối tượng cán bộ
```

Mảng đối tượng (tt) – Ví dụ

- Cài đặt chương trình thực hiện các yêu cầu:
 - Cài đặt lớp SinhVien (sinh viên) bao gồm các thuộc tính: Mã sinh viên, họ và tên, ngày sinh, giới tính, ngành học, điểm tổng kết và các phương thức cần thiết.
 - Cài đặt các chức năng:
 - Nhập vào một danh sách mới gồm n sinh viên.
 - Hiển thị danh sách n sinh viên vừa nhập ra màn hình.
 - Hiển thị ra màn hình thông tin của những sinh viên có điểm tổng kết cao nhất.

2.10. Con trỏ đối tượng

- Khai báo con trỏ đối tượng

`<Tên_Lớp> *<tên_con_trỏ>;`

- Trong đó:

- `<Tên_Lớp>`: Lớp chứa đối tượng mà con trỏ sẽ trỏ tới.
- `<tên_con_trỏ>`: đặt theo quy ước đặt tên trong C++.

- Ví dụ: `Nguoì nguoi, *p;`

- Cho con trỏ `p` trỏ tới đối tượng `nguoi` bằng lệnh gán:

`p = &nguoi;`

- Khi con trỏ `p` đã trỏ tới `nguoi`, có thể dùng con trỏ `p` để truy xuất đến các thành phần của đối tượng `nguoi`.
- Khi đó ta viết:

`<tên_con_trỏ>-><tên_thành_phần>;`

Con trỏ đối tượng (tt)

- Hai cách viết như sau là tương đương

Đối tượng nguoi thuộc lớp Nguoi	Con trỏ p chứa địa chỉ của nguoi
nguoi .hoTen	p-> hoTen
nguoi .ngaySinh	p-> ngaySinh
nguoi .nhap()	p-> nhap()
...	...

Con trỏ đối tượng (tt) – Ví dụ

```
class Ngnoi{
    private:
        char hoTen[30];
        int tuoi;
        char que[60];
    public:
        void nhap() {...}
        void xuat() {...}
};

int main(){
    Ngnoi nguoi, *p;
    p = &nguoi;
    p->nhap();
    p->xuat();
}
```

Con trỏ đối tượng (tt) – Con trỏ this

- **this** – con trỏ ngầm định: Là một con trỏ, được tự động tạo ra và luôn trỏ vào đối tượng hiện thời (đối tượng đang thực thi trong chương trình).
- Xét ví dụ:

```
class HìnhChuNhat {  
    private:  
        double chieuDai, chieuRong;  
    public:  
        void nhap() {  
            cout<<"Chieu dai: "; cin>>chieuDai;  
            cout<<"Chieu rong: "; cin>>chieuRong;  
        }  
};
```

Con trỏ đối tượng (tt) – Con trỏ this

- Thực chất sẽ viết là

```
class HìnhChuNhat {  
    private:  
        double chieuDai, chieuRong;  
    public:  
        void nhap() {  
            cout<<"Chieu dai: ";  
            cin>>this->chieuDai;  
            cout<<"Chieu rong: ";  
            cin>>this->chieuRong;  
        }  
};
```

2.11. Phương thức khởi tạo

- Khởi tạo giá trị ban đầu cho các thuộc tính của đối tượng mới được sinh ra.
- Cú pháp:

```
<Tên_Lớp> ([danh sách các đối])  
{  
    //Nội dung của phương thức  
}
```

Phương thức khởi tạo (tt) – Ví dụ 1

```
class PhanSo
{
    private:
        int tuSo, mauSo;
    public:
        //phương thức khởi tạo cho lớp PhanSo
        PhanSo()
        {
            tuSo = 0;
            mauSo = 1;
        }
};
```

Phương thức khởi tạo (tt)

- **Đặc điểm của phương thức khởi tạo:**
 - Một lớp có thể có phương thức khởi tạo, có thể không.
 - Phương thức khởi tạo thường được viết khi ta muốn khởi tạo các giá trị ban đầu cho các thuộc tính của đối tượng khi khai báo đối tượng.
 - Tên phương thức khởi tạo phải trùng với tên lớp.
 - Phương thức khởi tạo không có kiểu trả về và cũng không có giá trị trả về.

Phương thức khởi tạo (tt)

- **Phương thức khởi tạo được chia làm hai loại**
 - **Phương thức khởi tạo không đối**
 - Để khởi tạo các giá trị mặc định cho các thuộc tính của đối tượng.
 - Trong ví dụ 1 là một phương thức khởi tạo không đối, khởi gán các giá trị mặc định cho hai thuộc tính tuSo bằng 0 và mauSo bằng 1.
 - **Phương thức khởi tạo có đối**
 - Để khởi gán các giá trị bất kỳ cho các thuộc tính. Các giá trị của các đối sẽ được gán vào các thuộc tính.

Phương thức khởi tạo (tt) – Ví dụ 2

```
class PhanSo {  
    private:  
        int tuSo, mauSo;  
    public:  
        //Các PT khởi tạo cho lớp PhanSo  
        PhanSo() // PT khởi tạo không đối  
        {  
            tuSo = 0;  
            mauSo = 1;  
        }  
        PhanSo(int t, int m) // PT khởi tạo có đối  
        {  
            tuSo = t;  
            mauSo = m;  
        }  
};
```

Phương thức khởi tạo (tt)

- Sử dụng phương thức khởi tạo
- Khai báo và khởi tạo giá trị các thuộc tính của đối tượng

`PhanSo p1, p2 (2, 3) ;`

- Khi đó:

`p1` là một đối tượng thuộc lớp `PhanSo` với các giá trị của thuộc tính `tuSo = 0` và `mauSo = 1`;

`p2` được khởi tạo giá trị: `tuSo = 2` và `mauSo = 3`.

Phương thức khởi tạo (tt)

- Nhận xét về cách sử dụng phương thức khởi tạo:
 - Nếu một lớp có nhiều phương thức khởi tạo thì phương thức khởi tạo không đối là phương thức khởi tạo mặc định.
 - Nếu lớp **PhanSo** chỉ có phương thức khởi tạo có đối thì khai báo **p1** sẽ báo lỗi.
 - Nếu lớp **PhanSo** chỉ có phương thức khởi tạo không đối thì khai báo **p2(2, 3)** sẽ báo lỗi.

Phương thức khởi tạo (tt) – Ví dụ 1

- Cài đặt chương trình thực hiện:
 - Cài đặt lớp PhanSo (phân số) với các thuộc tính tuSo (tử số), mauSo (mẫu số) là các số nguyên và các phương thức:
 - Phương thức khởi tạo không đối khởi gán các giá trị mặc định tuSo = 0 và mauSo = 1.
 - Phương thức khởi tạo có đối khởi gán các giá trị bất kỳ cho tuSo và mauSo.
 - Phương thức xuất: in ra màn hình phân số dưới dạng Tử số/ Mẫu số.
 - Viết hàm main() khai báo và khởi tạo các phân số, in giá trị của các phân số ra màn hình.

Phương thức khởi tạo (tt) – Ví dụ 1

```
class PhanSo{
    private:
        int tuSo, mauSo;
    public:
        PhanSo() {
            tuSo = 0; mauSo = 1;
        }
        PhanSo(int t, int m) {
            tuSo = t; mauSo = m;
        }
        void xuat() {
            cout<<"Phan so: "<<tuSo<<"/"<<mauSo<<endl;
        }
};

int main() {
    PhanSo p1, p2(2,5);
    p1.xuat();
    p2.xuat();
}
```

Phương thức khởi tạo (tt) – Ví dụ 2

- Cài đặt chương trình thực hiện:
 - Thiết kế một lớp **HocSinh** bao gồm các thuộc tính: Họ tên, tuổi, điểm toán, điểm lý, điểm hoá.
 - Và các phương thức:
 - Phương thức khởi tạo không đối khởi tạo giá trị mặc định cho các thuộc tính của học sinh.
 - Phương thức khởi tạo có đối khởi tạo bộ giá trị bất kỳ cho các thuộc tính của học sinh.
 - Chương trình chính khởi tạo thông tin cho một học sinh, in thông tin học sinh lên màn hình.

2.12. Bài tập - 01

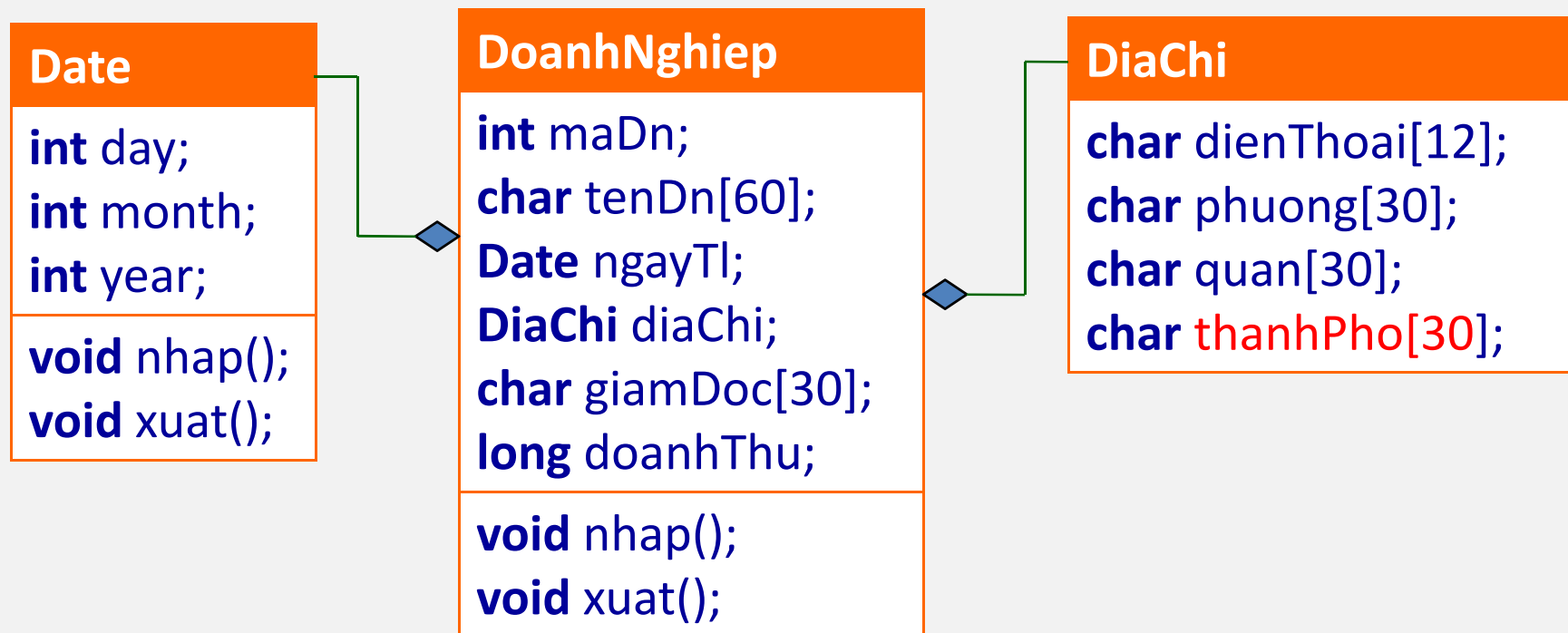
- Cài đặt chương trình thực hiện các yêu cầu:
 - Thiết kế một lớp HoaDon (Hoá đơn xuất hàng) bao gồm các thuộc tính: Mã hóa đơn, đơn vị nhận hàng, số tiền, người thanh toán, người nhận, ngày thanh toán, danh sách các hàng hóa (mỗi hàng hóa gồm mã hàng, tên hàng, số lượng, giá bán) và các phương thức cần thiết.
 - Cài đặt các chức năng:
 - Nhập vào thông tin của một hóa đơn,.
 - Xuất thông tin của hoá đơn lên màn hình.
 - Cho biết tổng tiền của hóa đơn.

Bài tập (tt) - 02

- Cài đặt chương trình thực hiện các yêu cầu
 - Cài đặt lớp XeHoi (xe hơi) gồm các thuộc tính: Nhãn hiệu, **hãng sản xuất**, kiểu dáng, màu sơn, năm sản xuất, xuất xứ, giá bán và các phương thức cần thiết.
 - Nhập vào một danh sách n xe hơi.
 - Hiển thị danh sách ra màn hình.
 - Hiển thị ra màn hình những xe hơi **của hãng “Toyota”**.
 - Sắp xếp danh sách theo chiều tăng dần của **giá bán**, in kết quả lên màn hình.

Bài tập (tt) - 03

- Cài đặt chương trình theo sơ đồ lớp sau:



Bài tập (tt) - 03

- Cài đặt các yêu cầu chức năng:

Nhập danh sách n doanh nghiệp.

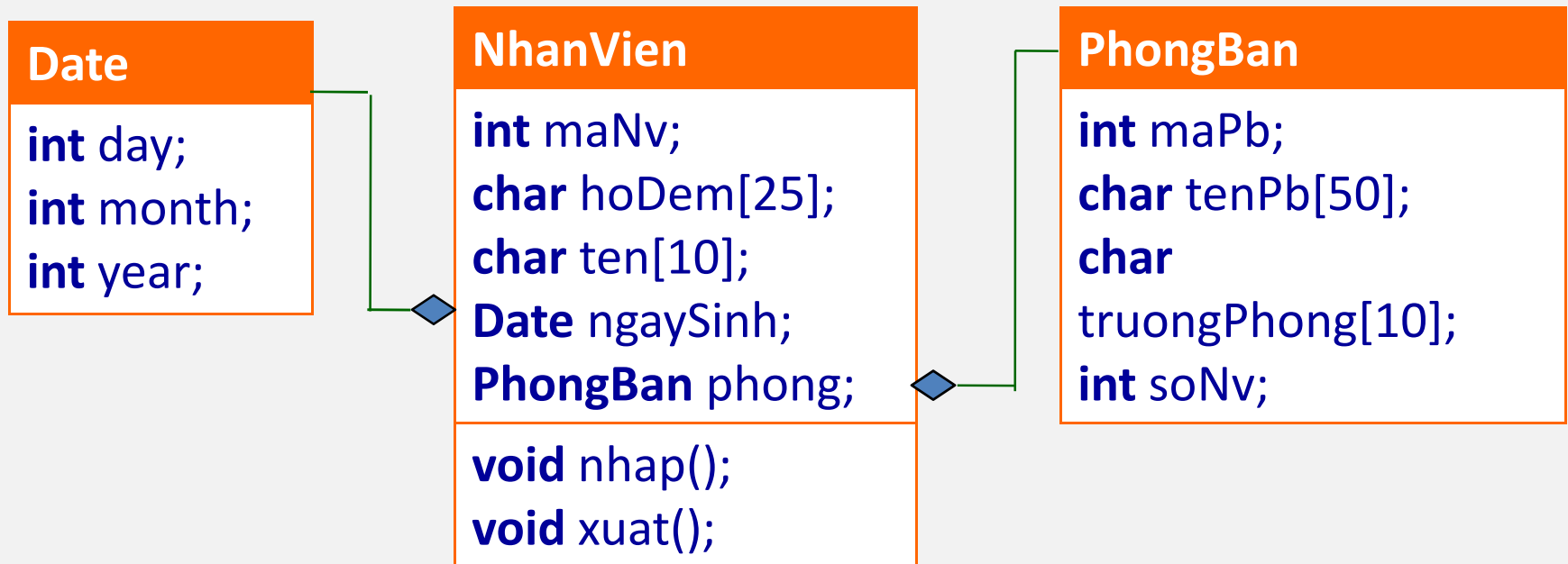
Hiển thị những doanh nghiệp ở thành phố Hà Nội ra màn hình.

Tính tổng doanh thu của những doanh nghiệp thành lập năm 2015.

Nhập vào mã của một doanh nghiệp, cho phép sửa lại toàn bộ thông tin của doanh nghiệp có mã vừa nhập (nếu có).

Bài tập (tt) - 04

- Cài đặt chương trình theo sơ đồ lớp sau:



Bài tập (tt) - 04

- **Cài đặt các yêu cầu chức năng:**

Nhập danh sách n nhân viên.

Hiển thị những nhân viên phòng tài chính ra màn hình.

Sắp xếp danh sách theo chiều tăng dần của tên nhân viên, hiển thị danh sách ra màn hình.

Nhập một nhân viên mới và số nguyên dương k, chèn nhân viên mới vào vị trí k trong danh sách.

Xóa nhân viên có mã 123.