C3_Friend function

Ngày học	@October 4, 2022
Property	Bai giang LTHDT - Chuong 3.pptx (phan 3).pdf
Property 1	

1. Định nghĩa hàm bạn

- Hàm bạn không phải là phương thức của lớp
- Việc truy cập tới hàm bạn được thực hiện như hàm thông thường.
- Trong thân hàm bạn của một lớp ***có thể truy cập tới các thuộc tính của đối tượng thuộc lớp*** này.
- Một hàm **có thể là bạn của nhiều lớp**.
- Hàm bạn được định nghĩa trong vùng **public**
- !!! Do hàm bạn không phải là phương thức của lớp nên tham số của hàm bạn **không có con trỏ this**

```
class A{
private : ...
public : ...
...
// Khai báo các hàm bạn của lớp A
friend void f1 (...) ;
friend double f2 (...) ;
...
} ;
// Xây dựng các hàm f1,f2
void f1 (...) //không có tên lớp
{ …}
double f2 (...)
{ …}
```

2. Thuộc tính tĩnh & phương thức tĩnh

a. Thuộc tính tĩnh

Thuộc tính tĩnh: khai báo với từ khóa **static**, được cấp phát 1 vùng nhớ cố định, tồn tại ngay cả khi class chưa có obj nào cả.

Thuộc tính tĩnh là **chung cho cả class**, không phải riêng của mỗi đối tượng

```
class A{
  public:
      static int x;
      int y;
};
A ob1, ob2;
```

ob1.y và ob2.y: có 2 vùng nhớ khác nhau

ob1.x và ob2.x: chỉ là 1, cùng biểu thị 1 vùng nhớ

Để sử dụng thuộc tính tĩnh, có thể dùng tên class, vd: A::x

Khai báo & gán giá trị cho thuộc tính tĩnh: thuộc tính tĩnh sẽ được cấp phát bộ nhớ và khởi gán giá trị bằng 1 câu lệnh khai báo ***đặt sau định nghĩa class***

```
int A::x = 0 // Khoi gan cho x = 0
```

Các thuộc tính của class ***không phụ thuộc vào obj*** nên khai báo là thuộc tính tĩnh

b. Phương thức tính tĩnh:

Được viết theo 1 trong 2 cách:

- Nếu xây dựng bên trong class, dùng từ khóa **static đặt trước định nghĩa phương thức** viết **bên trong định nghĩa class**
- Nếu xây dựng bên ngoài class, dùng từ khóa **static đặt trước định nghĩa phương thức** viết **bên trong định nghĩa class. KHÔNG CHO PHÉP dùng từ khóa static đặt trước định nghĩa phương thức** viết **bên ngoài class**

Các đặc tính:

- Là chung cho toàn bộ class, không lệ thuộc vào 1 obj cụ thể, tồn tại ngay cả khi class chưa có obj nào cả.

- Lời gọi phương thức tính tĩnh:

```
Tên class::tên phương thức tĩnh(các tham số thực sự)
```

- Vì phương thức tĩnh là độc lập với các đối tượng, nên phương thức tính tĩnh chỉ có thể truy nhập vào thuộc tính tĩnh.

3. Hàm tạo (constructor)

Hàm tạo mặc định là hàm tạo không có tham số.

Khi xây dựng hàm tạo cần lưu ý những đặc tính sau của hàm tạo:

- **Tên hàm tạo trùng với tên của lớp**.
- Hàm tạo **không có kiểu trả về.**
- Hàm tạo phải được **khai báo trong vùng public.**
- Hàm tạo có thể được xây dựng bên trong hoặc bên ngoài định nghĩa lớp.
- Hàm tạo có thể có đối số hoặc không có đối số.
- Trong một lớp có thể có nhiều hàm tạo (cùng tên nhưng khác các đối số). Dựa vào các tham số trong khai báo mà chương trình dịch sẽ biết cần gọi đến hàm tạo nào
- Nếu trong lớp đã có **ít nhất một hàm tạo**, thì hàm tạo mặc định sẽ không được phát sinh nữa. Khi đó mọi câu lệnh xây dựng đối tượng mới đều sẽ gọi đến một hàm tạo của lớp. Nếu không tìm thấy hàm tạo cần gọi thì chương trình dịch sẽ báo lỗi

```
class DIEM{
private:
int x,y;
public:
DIEM() //Ham tao khong tham so
{
x = y = 0;
}
DIEM(int x1, int y1) //Ham tao co 2 tham so
{
x = x1;
y=y1;
}
```

```
//Cac thanh phan khac
};
```

Hàm tạo sao chép

Là thành phần của lớp làm nhiệm vụ khởi tạo đối tượng bằng cách dùng 1 đối tượng khác cùng lớp đó

Tên class (const tên class & old_obj)

4. Hàm hủy (destructor)

Hàm huỷ là một hàm thành phần của lớp, có chức năng ngược với hàm tạo. Hàm huỷ được gọi để giải phóng một đối tượng trước khi đối tượng được hủy bỏ.

Nếu trong lớp không định nghĩa hàm huỷ thì một hàm huỷ mặc định không làm gì cả được phát sinh.

Trường hợp cần xây dựng hàm hủy thì tuân theo quy tắc sau:

- Mỗi lớp chỉ có một hàm hủy.
- Hàm hủy không có kiểu, không có giá trị trả về và không có đối số.
- Tên hàm hàm huỷ có một dấu ngã (~) ngay trước tên lớp.
- Ví dụ: ~DIEM() { }