LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Chương 3: Lớp và đối tượng (p2)

Trình bày: ThS. Lê Thanh Trọng

NỘI DUNG

1. Phạm vi truy xuất
2. Con trỏ this
3. Phép gán đối tượng
4. Hàm thiết lập – Constructor
5. Hàm thiết lập sao chép – Copy constructor

NỘI DUNG

1. **Phạm vi truy xuất**
2. Con trỏ this
3. Phép gán đối tượng
4. Hàm thiết lập – Constructor
5. Hàm thiết lập sao chép – Copy constructor

Phạm vi truy xuất

- Trong định nghĩa của lớp ta có thể xác định khả năng truy xuất thành phần của một lớp nào đó từ bên ngoài phạm vi lớp
- private, *protected* và public là các từ khoá xác định phạm vi truy xuất
- Mọi thành phần được liệt kê trong phần public đều có thể truy xuất trong bất kỳ hàm nào
- Những thành phần được liệt kê trong phần private chỉ được truy xuất bên trong ***phạm vi lớp (và hàm bạn, lớp bạn)***.

Phạm vi truy xuất

- Trong lớp có thể có nhiều nhãn private và public
- Mỗi nhãn này có phạm vi ảnh hưởng cho đến khi gặp một nhãn kế tiếp hoặc hết khai báo lớp.
- Nhãn private đầu tiên có thể bỏ qua vì C++ ngầm hiểu rằng các thành phần trước nhãn public đầu tiên là private

Phạm vi truy xuất – Ví dụ

```
class TamGiac{
  private:
    float a,b,c;/*độ dài ba cạnh*/
  public:
    void Nhap();/*nhập vào độ dài ba cạnh*/
    void Xuat();/*in ra các thông tin liên quan đến tam giác*/
  private:
    int LayLoai();//cho biết kiểu của tam giác: 1-d,2-vc,3-c,4-v,5-t
    float TinhDienTich();/*tính diện tích của tam giác*/
  };
```

Phạm vi truy xuất – Ví dụ

```
class TamGiac{
  private:
    float a,b,c;/*độ dài ba cạnh*/
    int LayLoai();//cho biết kiểu của tam giác: 1-d,2-vc,3-c,4-v,5-t
    float TinhDienTich();/*tính diện tích của tam giác*/
  public:
    void Nhap();/*nhập vào độ dài ba cạnh*/
    void Xuat();/*in các thông tin liên quan đến tam giác*/
};
```

Tham số hàm thành phần

```
void Point::KhoiTao (int xx, int yy){
        x = xx;
        y = yy;  //x, y la thanh phan cua lop Point
}
```

❖Hàm thành phần có quyền truy nhập đến các thành phần private của đối tượng gọi nó

Tham số hàm thành phần

- Hàm thành phần có quyền truy cập đến tất cả các thành phần private của các đối tượng, tham chiếu đối tượng hay con trỏ đối tượng có cùng kiểu lớp khi được dùng là tham số hình thức của nó

Tham số hàm thành phần

```
int KiemTraTrung(Point pt){
        return (x==pt.x && y==pt.y);
}
int KiemTraTrung(Point *pt){
        return (x==pt→x && y==pt→y);
}
int KiemTraTrung(Point &pt) {
        return (x==pt.x && y==pt.y);
}
```

?

NỘI DUNG

1. Phạm vi truy xuất
2. **Con trỏ this**
3. Phép gán đối tượng
4. Hàm thiết lập – Constructor
5. Hàm thiết lập sao chép – Copy constructor

Con trỏ this

- Từ khoá this trong định nghĩa của các hàm thành phần lớp dùng để xác định địa chỉ của đối tượng dùng làm tham số ngầm định cho hàm thành phần
- Con trỏ this tham chiếu đến đối tượng đang gọi hàm thành phần
- Ví dụ:

```
int KiemTraTrung(Point pt){
        return (this → x == pt.x && this → y == pt.y);
}
```

Con trỏ this

- Giúp phân biệt với tham số tham số truyền vào

```
int KiemTraTrung(int x, int y){
        return (this → x == x && this → y == y);
}
```

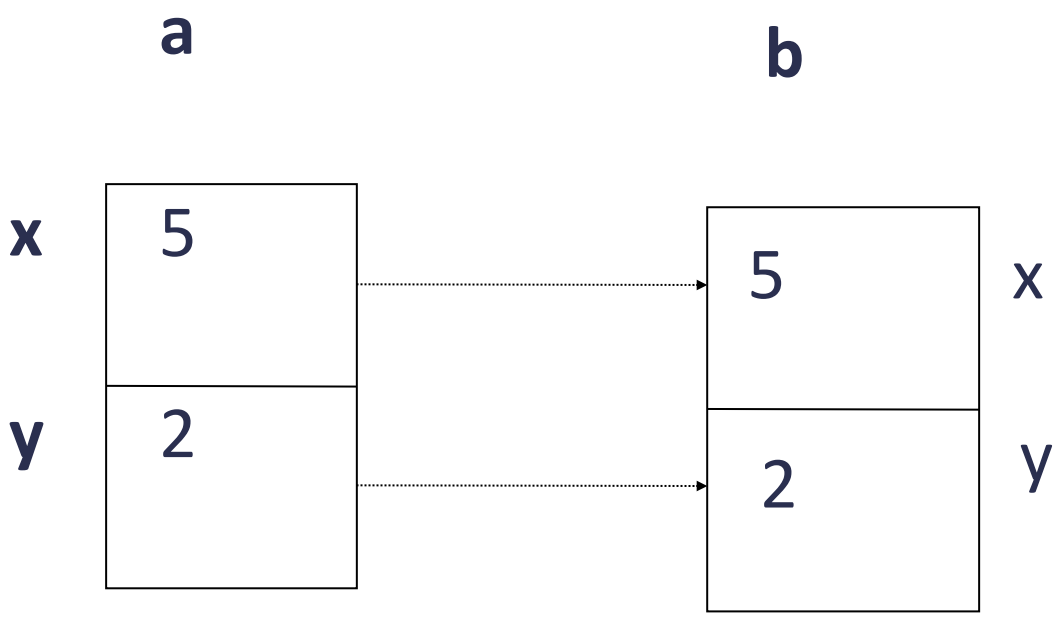
NỘI DUNG

1. Phạm vi truy xuất
2. Con trỏ this
3. **Phép gán đối tượng**
4. Hàm thiết lập – Constructor
5. Hàm thiết lập sao chép – Copy constructor

Phép gán đối tượng

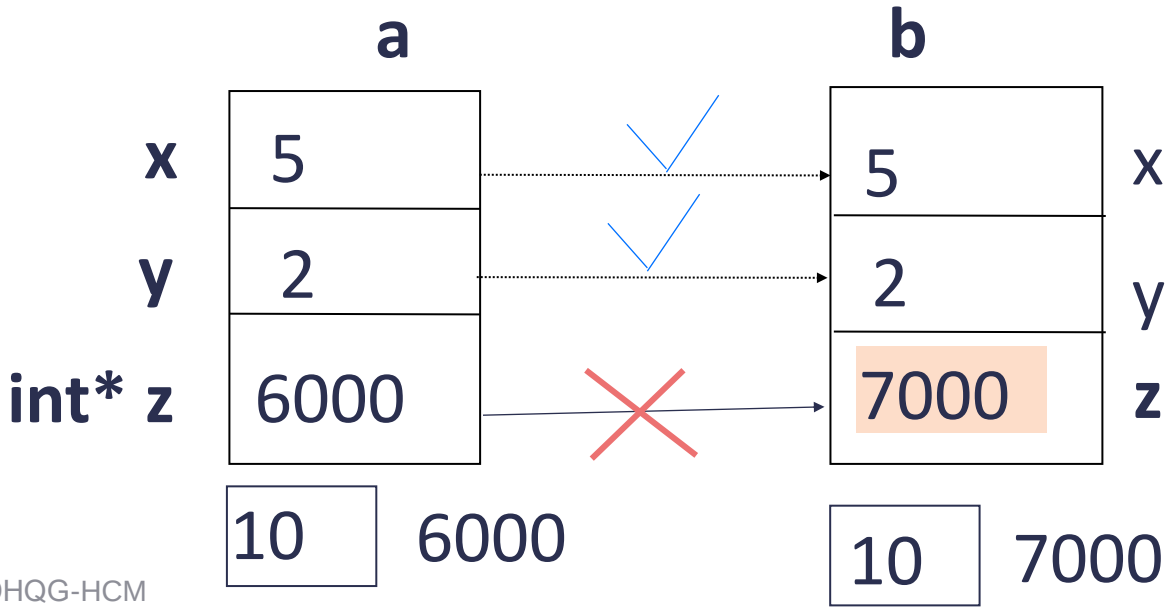
- Là việc sao chép giá trị các thành phần dữ liệu từ đối tượng a sang đối tượng b tương ứng từng đôi một
- Ví dụ:

```
Point a, b;
a.KhoiTao(5,2);
b = a;
```



Phép gán đối tượng

- Bình thường sử dụng phép gán có sẵn của trình biên dịch (dành cho các kiểu dữ liệu cơ bản, không phải con trỏ)
- Nếu lớp có chứa thành phần là con trỏ thì phải xây dựng lại phép gán cho lớp sao cho phù hợp




```
b.z = new int;
*b.z = * a.z;
```

NỘI DUNG

1. Phạm vi truy xuất
2. Con trỏ this
3. Phép gán đối tượng
4. **Hàm thiết lập – Constructor**
5. Hàm thiết lập sao chép – Copy constructor

Hàm thiết lập – Constructor

- Trong hầu hết các thuật giải, để giải quyết một vấn đề → thường phải thực hiện các công việc:
  - Khởi tạo giá trị cho biến, cấp phát vùng bộ nhớ của biến con trỏ, mở tập tin để truy cập,…
  - Hoặc khi kết thúc, chúng ta phải thực hiện quá trình ngược lại như: Thu hồi vùng bộ nhớ đã cấp phát, đóng tập tin,…
- Các ngôn ngữ OOP có các phương thức để thực hiện công việc này một cách *"tự động"* gọi là *phương thức thiết lập* và *phương thức hủy bỏ*

Hàm thiết lập – Constructor

- Constructor là một loại phương thức đặc biệt dùng để khởi tạo thể hiện của lớp
- Bất kỳ một đối tượng nào được khai báo đều phải sử dụng một hàm thiết lập để khởi tạo các giá trị thành phần của đối tượng
- Hàm thiết lập được khai báo giống như một phương thức với tên phương thức trùng với tên lớp và không có giá trị trả về (kể cả void)
- Constructor phải có phạm vi là public

Hàm thiết lập – Constructor

- Constructor có thể được khai báo chồng như các hàm C++ thông thường

- Constructor có thể được khai báo với các tham số có giá trị ngầm định (tham số mặc nhiên)

Ví dụ constructor

```
class Point{
    /*Khai báo các thành phần dữ liệu*/
    int x, y;
    public:
        Point() { x = 0; y = 0; }                    /*Hàm thiết lập mặc định*/
        Point(int ox, int oy) { x = ox; y = oy; }    /*Gán giá trị*/
        void DiChuyen(int dx, int dy);
        void Xuat();
};
Point a(5,2);       //ok?
Point b;            //ok?
Point c(3);         //ok?
```

Ví dụ constructor

```
class Point{
    /*Khai báo các thành phần dữ liệu*/
    int x, y;
    public:
        Point() { x = 0; y = 0; }                    /*Hàm thiết lập mặc định*/
        Point(int ox, int oy = 1){ x = ox; y = oy;}  /*Hàm thiết lập*/
        void DiChuyen(int dx, int dy);
        void Xuat();
};
Point a(5,2);      //ok?
Point b;           //ok?
Point c(3);        //ok?
```

Constructor mặc định

- Constructor mặc định (default constructor) là constructor được gọi khi thể hiện được khai báo mà không có đối số nào được cung cấp
  - MyClass x;
  - MyClass* p = new MyClass();
- Ngược lại, nếu tham số được cung cấp tại khai báo thể hiện, trình biên dịch sẽ gọi constructor khác (overload)
  - MyClass x(5);
  - MyClass* p = new MyClass(5);

Constructor mặc định

- Đối với constructor mặc định, nếu ta không cung cấp bất kỳ constructor nào, C++ sẽ tự sinh constructor mặc định là một phương thức rỗng.
- Tuy nhiên, nếu ta không định nghĩa constructor mặc định nhưng lại có các constructor khác, trình biên dịch sẽ báo lỗi không tìm thấy constructor mặc định nếu ta không cung cấp tham số khi tạo thể hiện.

Ví dụ constructor mặc định

```
class Point{
    /*Khai báo các thành phần dữ liệu*/
    int x, y;
    public:
            Point(int ox, int oy = 1){ x = ox; y = oy;}
           void DiChuyen(int dx, int dy);
           void Xuat();
};
Point a(5,2);      //ok?
Point b;           //ok?
Point c(3);        //ok?
```

NỘI DUNG

1. Phạm vi truy xuất
2. Con trỏ this
3. Phép gán đối tượng
4. Hàm thiết lập – Constructor
5. **Hàm thiết lập sao chép – Copy constructor**

Copy constructor

- Giúp tạo đối tượng mới "giống" đối tượng cũ một số đặc điểm, không phải hoàn toàn như phép gán bình thường
- Có tham số là tham chiếu đến đối tượng thuộc chính lớp này
- Có thể ta chỉ sử dụng một số thành phần nào đó của đối tượng ta tham chiếu → "gần giống nhau"

Copy constructor

- Nếu lớp không có các thành phần con trỏ có thể sử dụng copy constructor mặc định của trình biên dịch
- Ngược lại phải xây dựng copy constructor cho phù hợp

```
class PhanSo
{
        int TuSo;
        int MauSo;
public:
        PhanSo(int a, int b)
        {
                TuSo = a;
                MauSo = b;
        }
        PhanSo(const PhanSo& ps)
        {
                TuSo = ps.TuSo;
                MauSo = ps.MauSo;
        }
        void Xuat()
        {
                cout << TuSo << "/" << MauSo;
        }
};
int main()
{
        PhanSo ps1(3, 4);
        PhanSo ps2(ps1);
        ps2.Xuat();
        return 0;
}
```

Ví dụ copy constructor

```
class DaThuc
{
    private:
    int Bac;
    float* HeSo;
    public:
    DaThuc(){}
    DaThuc(int, float);
    DaThuc(const DaThuc&);
    void NhapDaThuc();
    void XuatDaThuc();
    ~DaThuc();
};
```

```
DaThuc::DaThuc(const DaThuc& DT)
{
    this->Bac = DT.Bac;
    this->HeSo = new float[Bac + 1];
    for (int i = Bac; i >= 0; i--)
        this->HeSo[i] = DT.HeSo[i];
}
```

Tóm tắt về lớp và đối tượng (p2)

❖Hàm thành phần (hay hàm thành viên)

- ❑Là các hàm thuộc lớp và có thể truy cập đến các thuộc tính (biến thành viên) và các hàm khác của lớp đó
- ❑Có thể được định nghĩa trực tiếp trong lớp hoặc bên ngoài lớp với từ khóa class_name::

❖Con trỏ this

- ❑Xác định địa chỉ của đối tượng dùng làm tham số ngầm định cho hàm thành phần
- ❑Tham chiếu đến đối tượng đang gọi hàm thành phần (this → x)

Tóm tắt về lớp và đối tượng (p2)

❖Phép gán đối tượng

- ❑Sử dụng phép gán **có sẵn** của trình biên dịch dành cho các kiểu dữ liệu cơ bản, không phải con trỏ
- ❑Nếu lớp có chứa thành phần là con trỏ thì phải **xây dựng lại** phép gán cho lớp sao cho phù hợp (sao chép dữ liệu, không sao chép địa chỉ)

Tóm tắt về lớp và đối tượng (p2)

❖Constructor

- ❑Là một loại phương thức đặc biệt dùng để khởi tạo thể hiện của lớp
- ❑Mục đích: khởi tạo giá trị cho biến, cấp phát vùng bộ nhớ của biến con trỏ, mở tập tin để truy cập,…
- ❑Bất kỳ một đối tượng nào được khai báo đều phải sử dụng một hàm thiết lập để khởi tạo các giá trị thành phần của đối tượng
- ❑Hàm thiết lập được khai báo giống như một phương thức với tên phương thức trùng với tên lớp và không có giá trị trả về (kể cả void)
- ❑Có thể được khai báo chồng, có thể có các tham số mặc nhiên
- ❑Constructor phải có phạm vi là public

Tóm tắt về lớp và đối tượng (p2)

❖Các loại constructor

❑Default constructor

- Không có đối số nào được cung cấp
- Ví dụ: MyClass x; hoặc MyClass* p = new MyClass();

❑Parameter constructor

- Có tham số được cung cấp tại khai báo thể hiện
- Trình biên dịch sẽ gọi constructor khác (overload)
- Ví dụ: MyClass x(5); hoặc MyClass* p = new MyClass(5);

❑Copy constructor

- Tạo đối tượng mới “giống” đối tượng cũ một số đặc điểm
- Không phải hoàn toàn như phép gán bình thường
- Có tham số là tham chiếu đến đối tượng thuộc chính lớp này: MyClass(const MyClass & mc)
- Ví dụ: MyClass x(5); MyClass y(x);

Bài tập về nhà

Bài 1: Xây dựng lớp Điểm (Point) trong hình học 2D

- Thuộc tính
  - Tung độ (float)
  - Hoành độ (float)
- Thao tác (phương thức)
  - Nhập, Xuất
  - Di chuyển theo vector (m,n)
  - Tính khoảng cách với một điểm

Viết chương trình nhập vào 2 điểm, xuất ra 2 điểm vùa nhập, khoảng cách giữa 2 điểm đó và tọa độ mới của 2 điểm khi tịnh tiến theo vector có giá trị do người dùng nhập vào.

Bài tập về nhà

Bài 2: Xây dựng lớp MangSoNguyen nhằm quản lý một mảng các số nguyên

- Thuộc tính
  - Số phần tử
  - Mảng động lưu giá trị của các phần tử là các số nguyên
- Thao tác (phương thức)
  - Nhập, Xuất
  - Thêm 1 phần tử vào cuối mảng
  - Xóa một phần tử
  - Tìm một phần tử
  - Chèn một phần tử vào vị trí bất kỳ (do người dung nhập vào)
  - Tính giá trị trung bình của mảng
  - Sắp xếp mảng tăng dần

Viết chương trình nhập vào một mảng và thực hiện kiểm tra các thao tác nêu trên.